

Generación de DTD para archivos XML utilizando una gramática visual relacional

Dr. Máximo López Sánchez	I.S.C Sandra Luz García Orta
<p>Centro Nacional de Investigación y Desarrollo Tecnológico (cenidet) Interior Internado Palmira S/N, Col Palmira Cuernavaca, Morelos CP 62490 <u>maximo@cenidet.edu.mx</u> <u>sandra-orta04c@cenidet.edu.mx</u> <u>www.cenidet.edu.mx</u> Tel (777) 3-18-77-41 Fax (777) 3-12-24-34 México</p>	

Resumen

Aunque el uso del lenguaje XML parece sencillo, en muchas ocasiones su misma simplicidad provoca que los desarrolladores olviden que hay otros estándares que lo apoyan y hasta lo complementan, como es el caso del DTD. Por lo que se ha caído en una mal práctica en el desarrollo de aplicaciones XML, en la cual la creación del DTD se deja de lado o se realiza posterior a la creación de la aplicación XML.

La consecuencia directa de esta forma de trabajo a llevado a crear soluciones que ayuden a los desarrolladores en la práctica de la creación de aplicaciones XML. Algunas soluciones propuestas parecen tan fáciles y simples que realmente no se puede ver que también son poco exactas. Como veremos la creación automática a partir de un XML restringe las etiquetas y estructura del DTD, limitando y hasta mal entendiendo la verdadera función que tiene un DTD.

Introducción

La evolución que se ha dado en los sistemas de información se a visto fuertemente apoyada por el gran desarrollo que a tenido el Internet. Así mismo la forma de presentar e intercambiar la información a través del Internet a requerido de la actualización de los protocolos de procesamiento de la información, en respuesta a esta demanda se ha creado el lenguaje de XML (lenguaje de marcas extensible) el cual cuenta con una gran aceptación hoy en día. Es evidente que la importancia de este relativamente nuevo lenguaje se debe a que a permitido darle "significado" a la información, además de estructurar y proporcionar una mejor forma de de controlar, compartir y aplicar esta sistematización a diferentes dominios de aplicación (comercial, industrial y personal).

Ya que las aplicaciones del lenguaje XML pueden variar o adaptarse según el dominio de aplicación, se hace necesaria la creación de los DTD (Document Type Definition, Definición de Tipos de Documentos), cuyo objetivo principal es permitir especificar y validar la estructura interna de un documento XML. Esto nos permite verificar que cada uno de los documentos XML que desarrollemos se encuentre dentro de la estructura que nosotros especificamos y que nos puede servir para estandarizar su uso a través de aplicaciones.

ROC&C'2005 - CP-79 PONENCIA RECOMENDADA
POR EL **COMITÉ DE COMPUTACIÓN**
DEL **IEEE SECCIÓN MÉXICO** Y
PRESENTADA EN LA **REUNIÓN DE OTOÑO, ROC&C'2005,**
ACAPULCO, GRO., DEL 29 DE NOVIEMBRE AL 4 DE DICIEMBRE DE 2005.

Desarrollo

▪ *XML Extensible Markup Language*

En la actualidad se ha creado la necesidad de intercambiar información entre las empresas, por lo que han buscado la manera de normalizar y comunicar dicha información.

Es así que hay diversas organizaciones encargadas de la elaboración de los estándares, dentro de las cuales sobresale la W3C al desarrollar el “XML (*Extensible Markup Language, Lenguaje de marcado extensible*), además de estar basado en la misma tecnología que HTML, esta diseñado para administrar mejor la información que el crecimiento de Internet requiere.”[1]

“XML es una metalenguaje, es decir, un lenguaje para la definición de lenguajes de marcado. XML definen la sintaxis y los requisitos que deben cumplir los lenguajes de marcado que especifica. XML en definitiva organiza y permite nuevas formas de incluir información extra.” [2]

Fue creado para organizar y presentar de manera estructurada datos dinámicos. Así mismo, para garantizar que estos datos se presenten de forma correcta de acuerdo a las reglas del contexto en el cual van a operar. La validación de un archivo XML se da a partir de otros tipos de archivos, ya que tiene asociados algunos estándares que lo complementan y que enriquecen la funcionalidad de XML, pero que disponen de entidad y personalidad propia, entre los que se encuentran los DTD, esquemas XML y DSD (Document Structure Description).

La forma más común de validar un archivo XML es a través de un archivo llamado DTD (*Document Type Definition*), en el cual se expresa la lógica de uno o más archivos XML. Otra forma de validar un archivo XML es a través de los esquemas XML, los cuales se basan en estructuras y tipos de datos para lograr validar cada elemento de un archivo XML.

Por lo tanto diremos que XML es un conjunto de reglas para definir etiquetas que nos

ayudan a organizar la información en documentos, entre sus objetivos destacan:

- Debe ser directamente utilizable en Internet.
- Debe soportar una gran variedad de aplicaciones.
- El diseño debe ser formal y conciso.

Los archivos XML deben de ser fácilmente creables.

▪ *Document Type Definition (DTD)*

“Un DTD, o Definición de Tipo de Documento, es un sistema de reglas ampliamente utilizado, con una peculiar sintaxis. Ya que debe definir las reglas para todos y cada uno de los elementos y atributos que aparecerán en el documento XML. De lo contrario, el documento no se considerará válido. Si en algún momento necesita añadir elementos al documento XML, deberá además agregar sus definiciones al DTD correspondiente o bien crear un nuevo DTD.”[3]

Por tanto, un DTD encierra una definición formal de un documento XML. Especifica su estructura lógica. Define tanto los elementos de una página como sus atributos. La inclusión de un DTD en un archivo XML es opcional, si no lo tiene, es un documento “bien formado” y si lo incluye es un documento “válido”.

Crear una definición de tipo de documento (DTD), es como crear nuestro propio lenguaje de marcado para una aplicación específica. Por ejemplo, podríamos crear un DTD que defina una tarjeta de visitas. A partir de ese archivo DTD, tendríamos una serie de elementos XML que nos permitirían definir tarjetas de visita. El DTD define los tipos de elementos, atributos y entidades permitidas, y puede expresar algunas limitaciones para combinarlos.

Los documentos que se ajustan a su DTD, se denominan "válidos". El concepto de "validez" no tiene nada que ver con el de estar "bien formado". Un documento "bien formado" simplemente respeta la estructura y sintaxis definida por la especificación de

archivos XML. Un documento "bien formado" puede además ser "válido" si cumple las reglas de un DTD determinado. Por lo que también existen documentos XML sin una DTD asociada, en este caso no son "válidos", pero tampoco "inválidos", simplemente se dice que son "bien formados".

Un ejemplo de DTD podría ser uno que nos defina un lenguaje de marcado para una base de datos de personas con direcciones de e-mail.

El archivo LISTADO.DTD podría ser algo así:

```
<?xml encoding="UTF-8"?>
<!ELEMENT lista (persona)+>
<!ELEMENT persona (nombre, email*, relación?)>
<!ATTLIST persona id ID #REQUIRED>
<!ATTLIST persona sexo (hombre | mujer)
#IMPLIED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT relación EMPTY>
<!ATTLIST relación amigo-de IDREFS #IMPLIED
enemigo-de IDREFS #IMPLIED> [4]
```

Basándonos en este DTD, podríamos escribir nuestro primer listado de un archivo XML de la siguiente manera:

```
<?xml version="1.0"?>
<!DOCTYPE listado SYSTEM "listado.dtd">
<lista>
  <persona sexo="hombre" id="ricky">
    <nombre>Ricky Martin</nombre>
    <email>ricky@puerto-rico.com</email>
    <relación amigo-de="Leticia">
      </persona>

  <persona sexo="mujer" id="Leticia">
    <nombre>Leticia Casta</nombre>
    <email>castal@micasa.com</email>
  </persona>
</lista>
```

▪ XML Schemas

“Un esquema, al igual que un DTD, define la “apariencia” de un conjunto de uno o más documentos XML, esto es, qué elementos y atributos pueden contener y en qué orden y cuál puede ser su contenido. De hecho, podríamos calificar los DTD como un tipo de esquema”. [5]

Los esquemas XML constan de dos partes: estructuras y tipos de datos. Básicamente esto se refiere a declarar un elemento con un tipo,

también es posible especificar que atributos y/o valores tendrá dicho tipo, además de poder especificar en que orden aparecerán dichos elementos, a esto se le llama una definición compleja de un tipo. La definición simple se refiere solamente a un conjunto de caracteres que se usaran como valores de atributos o datos de carácter.

Se especifica que dos elementos o dos tipos no se pueden definir con el mismo nombre, pero la declaración de un elemento y la definición de un tipo pueden usar el mismo nombre.

Un elemento en un documento XML es *válido* de acuerdo a un esquema XML establecido, si las reglas del tipo del elemento son satisfechas. Si todos los elementos son válidos, todo el *documento* es válido (a diferencia de los DTD's, no se requiere un elemento raíz específico). [6]

▪ Document Structure Description (DSD)

Los DSD (Document Structure Description) están diseñados para contener pocas y simples construcciones basadas en conceptos familiares, así como de un fácil entendimiento (aún por los que no sean expertos en XML). Además, cuentan con mucho más poder expresivo que otros lenguajes esquemáticos y con usos más prácticos. [7]

▪ ¿Por qué usar DTD's?

XML provee una manera de compartir los datos independiente de la aplicación. Con un DTD, grupos independientes de personas pueden ponerse de acuerdo en utilizar un DTD común para intercambiar información. Tu aplicación puede utilizar un DTD estándar para verificar que los datos que recibes son válidos. También puedes usar un DTD propio para verificar tus propios datos.

Un gran número de foros en internet están emergiendo para definir DTD's estándar para casi cualquier área de intercambio de datos.

Esto es, provee la tranquilidad que, dentro de un grupo de trabajo de un proyecto, todos los miembros intercambien información utilizando ciertas reglas y estructuras

comunes. Ya que de no ser así es posible que los trabajos individuales no pueden juntarse y esto equivale a tiempo y dinero perdido en realizar cambios y adecuarlos a cada uno.

▪ *Ejemplo práctico*

A continuación se verá un ejemplo práctico que muestra el problema que hay actualmente.

1. Supongamos que hemos creado el siguiente archivo XML (ver figura 3), el cual tiene como objetivo, describir la representación de un diagrama, que forma parte de un proceso de manufactura de nivel 1 denominado “Integración de componentes”.¹

```
<?xml version="1.0"?>
<diagrama>
  <componente id="c1" entrada="Materia prima" salida="Producto intermedio"/>
  <componente id="c2" entrada="Producto semiterminado" salida="Producto terminado"/>
  <funcionfisica id="f3" entrada="Producto intermedio" salida="Producto semiterminado"/>
  <siguiente>
    <x id="c1"/>
    <y id="f3" a="entrada"/>
  </siguiente>
  <siguiente>
    <x id="f3" a="salida"/>
    <y id="c2"/>
  </siguiente>
</diagrama>
```

Figura 1. Archivo XML de un diagrama del nivel 1 de un proceso de manufactura

Cuya representación visual se muestra en la figura 2:

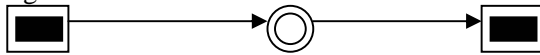


Figura 2. Representación visual del archivo XML de la figura 3

2. Una vez obtenido el archivo XML se procede a crear el archivo de validación, para mostrar la creación del archivo de validación en este caso concreto, utilizaremos un archivo DTD; la generación de archivo DTD, se puede lograr actualmente utilizando alguna de las herramientas comerciales disponibles en el mercado (por ejemplo, XMLWriter, EditML Pro) o creándolo en un editor de texto simple a mano. El archivo DTD resultante podría verse como en la figura 3.

```
<!-- DTD escrito a mano para validar el archivo XML del nivel 1
del proceso de manufactura "Integración de Componentes"
-->
<ELEMENT diagrama (componente, componente, funcionfisica, siguiente, siguiente) >

<ELEMENT funcionfisica EMPTY>
<ATTLIST funcionfisica id ID #REQUIRED>
<ATTLIST funcionfisica entrada CDATA #REQUIRED>
<ATTLIST funcionfisica salida CDATA #REQUIRED>

<ELEMENT componente EMPTY>
<ATTLIST componente id ID #REQUIRED>
<ATTLIST componente entrada CDATA #REQUIRED>
<ATTLIST componente salida CDATA #REQUIRED>

<ELEMENT siguiente (x, y)>

<ELEMENT x EMPTY>
<ATTLIST x id IDREF #REQUIRED>
<ATTLIST x a CDATA#FIXED "salida">

<ELEMENT y EMPTY>
<ATTLIST y id IDREF #REQUIRED>
<ATTLIST y a CDATA#FIXED "entrada">
```

Figura 3. DTD creado para el archivo XML de la figura 1

3. Si se realizara una validación del archivo XML de la figura 1 con el archivo DTD de la figura 3, observaríamos que es totalmente válido, ya que fue creado a partir del Archivo XML que esta validando, lo malo de crear de esta forma el archivo de validación es que si la aplicación XML estaba mal estructurada el archivo de validación no detectara dicho error.

4. Ahora supongamos que además se quiere representar el diagrama de la figura 4 en XML.

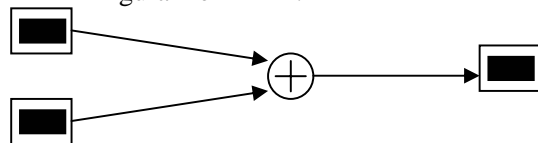


Figura 4. Otro diagrama de un proceso de manufactura del nivel 1

5. Bajo este nuevo diagrama, el archivo XML resultante podría quedar como en la figura 5.

```
<?xml version="1.0"?>
<diagrama>
  <componente id="c1" entrada="Materia prima 1" salida="Producto intermedio"/>
  <componente id="c2" entrada="Materia prima 2" salida="Producto intermedio"/>
  <componente id="c3" entrada="Producto intermedio" salida="Producto terminado"/>
  <ensamble id="e3" entrada="Materia prima" salida="Producto intermedio"/>
  <siguiente>
    <x id="c1"/>
    <y id="e3" a="entrada"/>
  </siguiente>
  <siguiente>
    <x id="c2"/>
    <y id="e3" a="entrada"/>
  </siguiente>
  <siguiente>
    <x id="e3" a="salida"/>
    <y id="c3"/>
  </siguiente>
</diagrama>
```

Figura 5. Archivo XML que representa el diagrama de la figura 4

¹ Para mayor información véase la referencia bibliográfica [11]

- Como se han incluido nuevos elementos y relaciones adicionales (véase figura 6) a las que se consideraron en el primer archivo XML de la figura 1, es imposible por parte del DTD previamente creado, el validar este nuevo archivo. Por lo tanto, sería necesario generar o elaborar un nuevo DTD basado en ambos casos.

```

<?xml version="1.0"?>
<!DOCTYPE diagrama SYSTEM "IntegracionA.dtd" >
<diagrama>
  <componente id="c1" entrada="Materia prima" salida="Producto intermedio"/>
  <componente id="c2" entrada="Producto semiterminado" salida="Producto terminado"/>
  <funcionfisica id="f3" entrada="Producto intermedio" salida="Producto semiterminado"/>
  <siguiente>
    <x id="c1"/>
    <y id="f3" a="entrada"/>
  </siguiente>
  <siguiente>
    <x id="f3" a="salida"/>
    <y id="c2"/>
  </siguiente>
</diagrama>

```

Primer archivo XML (figura 1)

sus propiedades. Valida archivos por DTD o Esquemas XML, automáticamente genera un DTD o Esquema a partir de un XML y automáticamente genera un XML de un DTD o Esquema. Importa información de una base de datos, un archivo de texto o incluso de hojas de Excel, automáticamente genera hojas de estilos XSL.[8]

Cuenta con la opción de preview para ver los

```

<?xml version="1.0"?>
<!DOCTYPE diagrama SYSTEM "Integracion.dtd" >
<diagrama>
  <componente id="c1" entrada="Materia prima 1" salida="Producto intermedio"/>
  <componente id="c2" entrada="Materia prima 2" salida="Producto intermedio"/>
  <componente id="c3" entrada="Producto intermedio" salida="Producto terminado"/>
  <ensamble id="e3" entrada="Materia prima" salida="Producto intermedio"/>
  <siguiente>
    <x id="c1"/>
    <y id="e3" a="entrada"/>
  </siguiente>
  <siguiente>
    <x id="c2"/>
    <y id="e3" a="entrada"/>
  </siguiente>
  <siguiente>
    <x id="e3" a="salida"/>
    <y id="c3"/>
  </siguiente>
</diagrama>

```

Segundo Archivo XML (figura 5) incluye respecto al primer archivo:

- Un componente adicional
- Una función nueva
- Una relación extra

Figura 6. Muestra la inclusión de nuevos elementos en el archivo XML de la figura 4

documentos con la transformación de los XSL. Utiliza búsqueda por XPath Visual.

- Por lo tanto, sería necesario generar o elaborar un nuevo DTD basado en ambos casos y así sucesivamente cada que exista una modificación en la estructura del archivo de aplicación XML.

Generación de DTD a partir de XML

Esta propuesta de trabajo de graduación es de la *Fundación Universitaria San Martín*, en Bogotá, Colombia. Ya está siendo desarrollada actualmente y como su nombre lo indica, genera código DTD a partir de código XML. [9]

Como se puede observar, resulta costoso y hasta cierto punto molesto el tener que estar generando archivos DTD conforme se van ajustando los elementos y situaciones a representar en los archivos XML.

▪ Estado del arte

EditML Pro

Es un software comercial elaborado por la compañía NetBryz Technologies (2000-2004), principalmente es un editor de archivos XML. Cuenta con una vista en forma de árbol para ver el documento XML, así como también tiene una vista del código fuente en la cual es posible utilizarlo totalmente en modo texto, de lo contrario debemos seleccionar un elemento de la vista de árbol y en la ventana de la derecha modificar la información referente a

Serialización en XML de objetos Java (SerXML)

Este proyecto de fin de carrera de la *Universidad Rey Juan Carlos en España*, durante el periodo 2003/2004. Es acerca de la serialización de archivos DTD y Schemas de XML a partir de clases en Java.[10]

Como podemos apreciar, actualmente existen ya herramientas en el mercado y otras que están en desarrollo que nos permiten el generar un archivo DTD. Sin embargo, en ningún caso los archivos DTD generados pueden en algún momento aplicarse de forma general a los archivos XML que pretenderán validar, ya que éstos son simplemente creados a partir de la especificación de archivo XML que van a validar. Por lo tanto, si en algún

momento la estructura del XML es extendida o modificada, el DTD no podrá seguir validando el archivo XML, la desventaja de crear los DTD a partir del XML es que los DTD quedan propietarios al XML del cual fueron creados.

Conclusiones

Como podemos observar de acuerdo a los hechos presentados, la necesidad de generar una consciencia de planeación adecuada de los contenidos en los documentos XML es importante y aunque se hayan realizado buenos trabajos con respecto a la generación automática partiendo como base del XML, lo mejor es que cada desarrollador defina su archivo de validación, llámese DTD, XML Schemas o DSD.

Una cultura de planeación ayudará a reducir el número de errores y el tiempo de desarrollo dando una forma más sencilla y práctica para solucionar este problema, aunque no por eso quiere decir que sea la única, es posible que en un futuro se preste mucha más atención a la creación automática de estos archivos, tanto para mejorar los generadores actuales como para crear nuevas técnicas o métodos al respecto, conforme vaya avanzando cada vez más la tecnología alrededor de XML.

Bibliografía

- [1] Castro Elizabeth. GUIA DE APRENDIZAJE XML. Editorial PEARSON EDUCACION, Madrid, 2001, pág. XIII.
- [2] Gutiérrez Abraham y Martínez Raúl. XML A TRAVES DE EJEMPLOS. Editorial Alfaomega, Colombia, 2001, pág. XVII.
- [3] Castro, op. cit., pág. 23.
- [4] Document Type Definition (DTD). (Alfredo Reino). Obtenido el 25 de mayo del 2005, de <http://www.ulpgc.es/otros/tutoriales/xml/DTD.html>
- [5] Castro, op. cit., pág. Pág. 51.
- [6] XML tutorial: Overview of XML Schema. (Anders Møller & Michael I. Schwartzbach). Obtenido el 1 de junio del 2005, de <http://www.brics.dk/~amoeller/XML/schemas/xmlschema-overview.html>

[7] XML tutorial: Document Structure Description 2.0. (Anders Møller & Michael I. Schwartzbach). Obtenido el 4 junio del 2005, de <http://www.brics.dk/~amoeller/XML/schemas/dsd-overview.html>

[8] Products ->EditML Pro – XML Editor. NetBryx Technologies. Obtenido el 15 de enero del 2005, de <http://www.netbryx.com/EditMLPro.aspx>

[9] Propuesta de Trabajo en graduación. Fundación Universitaria San Martín, Bogota. Obtenido el 12 de enero del 2005, de <http://ingenieria.sanmartin.edu.co/~itamara/temastesis/>

[10] SerXML – Serialización en XML de objetos Java. (David Arbona Navarro). Universidad Rey Juan Carlos. Obtenido el 15 de enero del 2005, de <http://vido.escet.urjc.es/serxml/documentation.htm>

[11] Isidro Moctezuma Cantorán. AMBIENTE VISUAL PARA EL DISEÑO Y MODELADO DE PROCESOS DE MANUFACTURA. Tesis del CENIDET. octubre 2004. Página 43.

Curriculum Vitae



Máximo López Sánchez. Es investigador en el Centro Nacional de Investigación y Desarrollo Tecnológico (cenidet). Sus áreas de interés son: Lenguajes Visuales, Ambientes Integrados, Reusabilidad de Código, Modelado de Sistemas, Sistemas en Tiempo Real y Sistemas Interactivos.



Sandra Luz García Orta, Ingeniero en Sistemas Computacionales egresado del Instituto Tecnológico de Cd. Victoria en el año 2004. Actualmente, estudiante de Maestría en Ciencias en Ciencias de la Computación, en el área de Ingeniería de Software del Centro Nacional de Investigación y Desarrollo Tecnológico (cenidet).