Arquitectura de la información: XML y WEB

José Vicente Rodríguez Muñoz
Departamento de Información y Documentación
Universidad de Murcia
jovi@um.es

Pedro Manuel Díaz Ortuño
Departamento de Informática y Sistemas
Universidad de Murcia
diazor@um.es

R E S U M

Els documents hipermèdia oberts es caracteritzen per un suport de marcatge estructural i depenent de l'àmbit d'aplicació, l'ús d'estructures d'enllaç comunes a diferents conjunts de documents i l'especificació de la presentació de la informació. XML permetrà transformar la web en un sistema obert, cobrint els quatre aspectes tradicionals dels sistemes hipermèdia: contingut, estructuració, presentació i navegació.
La necessitat d'interoperativitat entre les diferentes aplicacions i plataformes ha endegat l'interès per l'estàndard XML, el qual permet descriure l'estructura dels documents amb independència tant de la presentació com del suport que conté les dades. A l'entorn de XML estan sorgint una família de llenguatges que se centren en algun tractament específic del document.
Paraules clau: Document electrònic, Estàndard, HTML, Llenguatges de marcat, SGML, Sistema hipermèdia, XML

R E S U M E N

Los documentos hipermedia abiertos se caracterizan por el soporte de marcado estructural y dependiente del ámbito de aplicación, el uso de estructuras de enlaces comunes a conjuntos de diferentes documentos y la especificación de la presentación de la información. XML permitirá transformar la web en un sistema abierto, abordando los cuatros aspectos tradicionales de los sistemas hipermedia: contenido, estructuración, presentación y navegación.
La necesidad de interoperabilidad de diferentes aplicaciones y plataformas, ha promovido el interés por el estándar XML. Éste permite describir la estructura de los documentos, con independencia tanto de su presentación como del soporte donde se asienten sus datos. En torno a XML está surgiendo toda una familia de lenguajes que se centran en algún tratamiento específico del documento.
Palabras clave: Documento electrónico, Estándar, HTML, Lenguajes de marcado, SGML, Sistema hipermedia, XML.

A B S T R A C T

The open hypermedia documents are characterized by the support of structural marking and depending on the application range, the use of common linking structures to groups of different documents, and the specification of the information presenta-

tion. XML will permit to transform the Web in an open system, dealing with the four traditional aspects of the hypermedia systems: content, structure, presentation and navigation.
The need for interoperability of different applications and platforms, has promoted an interest for the standard XML, which allows to describe the structure of documents, independently of their presentation as of the place where the data is stored. All around XML an entire range of languages is appearing centered on some specific treatment of a document.
Keywords: Electronic document, Hypermedia system, HTML, Marking languages, SGML, Standard, XML.

1. Introducción

Un sistema hipermedia se caracteriza por organizar los datos en grafos de nodos y donde el acceso se realiza por medio de enlaces entre los nodos. De esta manera, el usuario obtiene la información «navegando» entre los nodos, utilizando la denominada metáfora *point and clic*. El web puede considerarse como un gigantesco sistema hipermedia compuesto por documentos enlazados. HTML es actualmente el idioma de la web, el lenguaje en que están escritos los documentos hipermedia que lo componen. Estos documentos responden al modelo de documento definido por HTML.

Desde una perspectiva orientada al documento, un sistema hipermedia abierto (OHS, Open Hypermedia System) se caracteriza por no basarse en un único modelo de documento hipermedia. Un OHS puede procesar un conjunto extensible de tipos de documentos, para reconocer la (posiblemente compleja) estructura de hiperenlaces de los mismos, y para presentar a éstos de la manera más apropiada al usuario. Desde esta perspectiva, la web no puede calificarse actualmente como un OHS, porque no pueden extenderse fácilmente los navegadores a nuevos tipos de documentos: contamos con medios muy limitados para decir al navegador cómo reconocer los enlaces codificados de forma diferente a los enlaces de HTML, o para definir cómo deben presentarse los nuevos tipos de documentos al usuario.

En contraste, los modelos de documentos hipermedia abiertos centran su atención en medios que soporten el marcado estructural y dependiente del dominio, medios para usar las estructuras de enlaces comunes a conjuntos de diferentes documentos, y maneras genéricas de especificar la presentación de la información, normalmente mediante hojas de estilo (OSSENBRUGGEN; et al., 1998).

Además de considerar a XML como el futuro idioma de la web, hay un segundo aspecto que consideramos interesante destacar: XML puede considerarse como un formato estándar de intercambio de información entre aplicaciones. XML es un lenguaje para la descripción de datos estructurados de forma que puedan ser compartidos por múltiples aplicaciones sobre diferentes plataformas. Permite a diferentes organizaciones usar distintos tipos de *software* y *hardware* para comunicarse efectivamente mediante un lenguaje que todos pueden hablar, eliminando las barreras a la interoperabilidad de aplicaciones.

Los sistemas de información de las organizaciones se han centrado tradicionalmente en sus procesos internos, gestionando las relaciones con agentes externos mediante documentos impresos (facturas, pedidos, etc). Gran parte de la información gestionada se encuentra en soportes heterogéneos *(*documentos generados por procesadores de texto*,* hojas de cálculo, páginas *HTML* o simples ficheros) y el resto es confiada a Sistemas de

Gestión de Bases de Datos tradicionales. El trasiego de información entre las diferentes aplicaciones requiere problemáticas conversiones entre formatos distintos y a menudo incompatibles.

Por otra parte, los *SGBD* imponen una disciplina en la estructuración de los datos que puede resultar excesivamente restrictiva para reflejar la variedad de tipos de documentos que maneja una organización. La gestión eficiente de los datos contenidos en estos sistemas ha conllevado una disposición de los datos en estructuras sencillas como la representación tabular. Si bien esta estructura facilita el tratamiento informático, no resulta adecuada para el tipo de información contenida en los documentos tradicionales. XML, y su *familia* de lenguajes (XSL, XQuery, etc.), permite describir más fielmente la información no estructurada de los documentos tradicionales. Estas características lo hacen adecuado para ser utilizado como *lingua franca* para la integración de aplicaciones y fuentes de datos heterogéneas. Ofrece la flexibilidad y ductilidad necesaria para adaptar e integrar los datos contenidos en los distintos soportes al formato esperado por la aplicación receptora.

2. Lenguajes de la web

Las iniciales XML provienen de *Extensible Markup Language*, es decir, «lenguaje de marcas extendible». Es un metalenguaje (lenguaje que describe los datos y como éstos se estructuran), mediante el cual los desarrolladores pueden crear sus propios elementos para alcanzar sus propias necesidades de información. XML se usa para crear meta-vocabularios (conjuntos de etiquetas usados para representar elementos dentro de un documento XML) adaptados a las necesidades de cada industria o disciplina. Cuando se definen etiquetas en un documento, se pueden crear vocabularios específicos, o conjunto finito de términos expresados como etiquetas, que describen los datos (DTD, Document Type Description, Descripción de tipo de documento).

Un documento XML contiene solamente datos y etiquetas. Su objetivo es separar contenido de presentación, permitiendo gran flexibilidad en el procesamiento y la visualización. La estructuración de datos usando XML permite:

- Acceso a datos a través de múltiples plataformas y aplicaciones.
- Realización de búsquedas eficientes.
- Utilización de diferentes «hojas de estilos» para la visualización de datos de diferentes formas.

En la comunidad web se tiende a considerar a XML como una versión ampliada de HTML, confundiendo el significado de «Extensible» y dando por hecho que XML es una ampliación de HTML. Para aclarar este error es imprescindible empezar hablando de los lenguajes de marcas.

2.1 Lenguajes de marcas

Cada sistema propietario utiliza sus propias marcas para describir las entidades de los documentos. Las «marcas» son los códigos que indican a la aplicación cómo debe tratar a su contenido. Si se desea que un texto aparezca en cursiva, cada aplicación introduce al principio y al final del texto correspondiente una marca que le permita mostrarlo en pantalla e imprimirlo adecuadamente. Lo mismo ocurre con las tablas, los márgenes, las imágenes, los tipos de letra, los enlaces, etc. Conocer las marcas que utiliza cada programa de trata-

miento de documentos hace posible diseñar filtros que permiten convertir la información de unos formatos de marcas a otros.

IBM intentó resolver los problemas asociados al tratamiento de documentos en diferentes plataformas a través de lo que denominó GML (*Generalized Markup Language*, lenguaje de marcas generalizado). La separación entre estructura y aspecto es lo que había motivado a IBM en un principio, y es a Goldfarb a quién debemos el término *etiquetado descriptivo (generalizado)* para describir este enfoque sobre la preparación de documentos. La idea del etiquetado generalizado era que cada etiqueta sirviese tanto para describir el aspecto exterior del texto (el formato), como para indicar su contenido (el tipo de información o dato). La solución adoptada utilizaba etiquetas de descripción de datos relacionadas con plantillas de estilos de formato.

Más tarde, GML se lo convirtió en un estándar oficial (ISO 8.879), denominándose SGML (*Standard Generalized Markup Language*, lenguaje de marcas generalizado estándar). Esta norma de carácter general se utiliza para diseñar lenguajes de marcas específicos. En 1989, Tim Berners-Lee aplicó las normas de SGML para diseñar HTML como herramienta de publicación de investigaciones y colaboración entre autores en el CERN.

Se pueden considerar tres utilizaciones básicas de los lenguajes de marcas: descripción del contenido, definición del formato y los que realizan las dos funciones indistintamente. Las aplicaciones de bases de datos son buenas referencias del primer sistema, los programas de tratamiento de textos son ejemplos típicos del segundo tipo, y HTML puede verse como el ejemplo más conocido del tercer modelo.

2.2 HTML

El lenguaje HTML es originariamente una aplicación de SGML, especializada en la descripción de documentos en pantalla a través de marcas (*tags*, etiquetas). HTML es un lenguaje de marcas diseñado para publicar documentos en la web con la máxima sencillez.

En principio, la intención de HTML era que las etiquetas fueran capaces de marcar la información de acuerdo con su significado. Era un lenguaje de marcas orientado a describir los contenidos, no la forma de presentación de los mismos en pantalla: el título del documento, los títulos de los apartados, el autor del documento, los textos resaltados, etc., eran marcados por las etiquetas TITLE, Hx, ADDRESS, STRONG, etc., dejando a cada visualizador (*browser*) la tarea de formatear el documento según su criterio.

Esto podía producir presentaciones diferentes, pero permitía controlar fácilmente el contenido. Además, se tiene la posibilidad de especificar el formato del documento con descripciones de formato particulares, como es el caso de las CSS (*Cascade Style Sheet,* hojas de estilo en cascada). Por diversos motivos, los navegadores fueron incorporando etiquetas HTML dirigidas a controlar la presentación (FONT, CENTER, COLOR, etc.), por lo que HTML pasó a ser un lenguaje de marcas cada vez más dirigido al control de la presentación.

Además, los analizadores sintácticos de los navegadores permitieron saltarse algunas normas (por ejemplo, trabajar sólo con la etiqueta de principio de párrafo <P>, sin la de final </P>), dando como resultado que HTML ya no es un lenguaje que sigue las normas estrictas del SGML.

Puesto que HTML se convirtió en una entidad que dejó de servir para su función inicial, el Consorcio World Wide web (W3C) inició la descripción de un nuevo subconjunto del SGML que sirviera para describir contenidos de documentos. Se ha denominado XML, y en 1998 han sido publicadas las especificaciones de la versión 1.0.

En la siguiente tabla se muestran algunas diferencias entre HTML y XML:

HTML	XML
Formatea texto para la visualización	Describe los datos y su estructura
Utiliza un pequeño conjunto fijo de etiquetas de formato	Estructura los datos con etiquetas específicas de usuario
No distingue entre mayúsculas y minúsculas	Distingue entre mayúsculas y minúsculas
No entiende la estructura de los datos	Es entendido tanto por humanos como por máquina
Ignora errores sintácticos	Aplica estrictas reglas sintácticas

2.3 XHTML

La «guerra» que han mantenido las principales empresas creadoras de los navegadores (Microsoft y Netscape) por imponer «sus etiquetas» (*tags*), ha dado lugar a lenguajes que no siguen las mismas normas y estándares. La mayoría de las páginas web existentes en Internet presentan código mezcla del estándar HTML y de las especificaciones particulares de los navegadores.

La existencia y utilización de etiquetas no especificadas por las normas y el consentimiento de «incorrecciones gramaticales» por los navegadores, generan una situación difícil de controlar. El W3C, aprovechando la inercia que ha provocado la publicación del estándar XML, mucho más estricto con las reglas del código, intenta terminar con parte del desajuste actual. Ha dictado reglas expresas para distinguir el HTML que sigue a rajatabla las normas de XML, denominándolo XHTML (*eXtensible HyperText Markup Language*) que no es más que una reformulación de HTML4 dentro de las normas de XML. Esta nueva norma describe las especificaciones que deben respetarse para generar un código estricto, ajustado a las reglas gramaticales que debe guardar una página web HTML bien confeccionada.

Por supuesto que esta normativa no resuelve todos los problemas de HTML, pero sí ayudará a eliminar los errores gramaticales, unificando la descripción del código y facilitando la portabilidad de los documentos.

Las razones esgrimidas por el W3C para aconsejar el uso del XHTML son dos, principalmente:

- XHTML, ya que es una aplicación XML, ha sido diseñado para ser extensible: se pueden añadir nuevas etiquetas o elementos a las descripciones de tipos de documento.

- XHTML ha sido diseñado pensando en la portabilidad: en los próximos años se producirá un aumento considerable de los dispositivos que traten información en código HTML. Televisiones, teléfonos móviles, ordenadores de bolsillo, calculadoras, hornos, etc., soportarán código HTML, siempre que esté realmente estandarizado y no requiera procesamientos complejos asociados a incumplimientos de la norma.

2.4 XML

Actualmente, HTML se usa como el lenguaje de marcado y enlace en la web. Desde su introducción, HTML ha sido criticado tanto por la comunidad SGML como por la comunidad hipermedia. Como ya se ha comentado, las preocupaciones de la comunidad SGML se centraron en la separación del marcado estructural y los problemas de estilo. Este problema era (en parte) resuelto por la introducción de las CSS, lenguaje de especificación de hojas de estilo, que permite a los autores separar la presentación del contenido. Sin embargo, la preocupación principal de la comunidad SGML es que HTML está constreñido a un único tipo de documento, que contrasta con los distintos tipos de documento SGML en uso dentro de la

comunidad SGML, cada uno de ellos a medida de una aplicación específica. SGML exige a las aplicaciones especificar formalmente los elementos a usar en una DTD (*Document Type Definition*, Definición de Tipo de Documento).

Para una distribución de documentos SGML en web tan sencilla como la de los documentos HTML, se diseñó XML como «un pequeño subconjunto de SGML» en el que han sido eliminados muchos de los rasgos más exóticos de SGML.

En la actual situación, en teoría, HTML es un subconjunto de XML especializado en presentación de documentos para la web, mientras que XML es un subconjunto de SGML especializado en gestión de información para la web.

La particularidad más importante del XML es que no posee etiquetas prefijadas con anterioridad, ya que es el propio diseñador el que crea o especifica el *vocabulario*, dependiendo del contenido del documento. De esta forma, los documentos XML con información sobre libros tienen etiquetas como <AUTOR>, <TITULO>, <EDITORIAL>, <ISBN>, etc., mientras que los documentos XML relacionados con educación incluyen etiquetas del tipo de <ASIGNATURA>, <ALUMNO>, <NOTA>, etc.

3. Estructura de los documentos

Puesto que los desarrolladores pueden crear su propio vocabulario, las organizaciones necesitan ser capaces de acordar estándares de estructuras de documentos para que sus miembros puedan «hablar» entre ellos. La especificación de la estructura de un documento XML separadamente del contenido permite el control de la consistencia de documentos creados por diferentes autores. Éste es el propósito de una DTD o un «esquema» en XML.

Una DTD especifica los elementos, atributos, entidades y relaciones permitidos en un documento XML. Una DTD puede estar incluida en el mismo documento que describe, o en un documento separado que podemos referenciar mediante su URL. Describe los datos, proporcionando la gramática y el vocabulario del lenguaje que se está usando en un documento XML. Mediante una DTD se asegura que los autores entiendan la estructura y vocabulario de los datos y los usuarios reciban toda la información que necesitan de forma consistente.

En XML no existen DTD predefinidas, por lo que es labor del diseñador especificar su propia DTD para cada tipo de documento XML. En la especificación de XML se describe la forma de definir DTD particularizadas para documentos XML, que pueden ser internas (cuando van incluidas junto al código XML) o externas (si se encuentran en un archivo propio).

Una de las diferencias más práctica entre XML y SGML es el hecho de que en XML la definición de tipo de documento es opcional. Esto permite documentos «ligeros» e implementaciones que pueden emplear la flexible aproximación de XML al marcado, sin la sobrecarga del análisis gramatical de DTD y de la validación de documentos.

3.1 Creación de DTD

Una DTD usa una sintaxis que no está basada en XML para definir elementos, atributos y relaciones. Esta sintaxis consiste en un pequeño conjunto de sentencias declarativas, un conjunto de símbolos que ayudarán a definir la estructura de datos, y palabras reservadas para especificar tipos de datos.

Los atributos proporcionan información a las aplicaciones que trabajan con los datos XML. Normalmente se definen a continuación del elemento con el que son usados, indicando el tipo (CDATA, ID, IDREF, ENTITY,…) y el valor por defecto.

Como ejemplo sencillo, para una DTD de libros se podrían establecer los siguientes elementos:

<!ELEMENT CATALOGO (LIBRO+)>	Declara el elemento raíz «CATALOGO» y a «LIBRO» como su hijo.
<!ATTLIST CATALOGO Tipo (informática | documentación) «informática»>	Declara el atributo «Tipo» de «CATALOGO», con la lista de posibles valores (tipo enumeración) y el valor por defecto.
<!ELEMENT LIBRO (Título, (Autor+ | Editor+), Editorial, Precio?) >	Declara el elemento «LIBRO» y a los elementos hijo que puede contener en el orden exacto en que los elementos pueden aparecer en el documento. La barra (|) indica una opción entre dos elementos.
<!ATTLIST LIBRO ISBN CDATA «000000»> <!ATTLIST LIBRO Año CDATA)	Declara los atributos «ISBN» y «Año» del elemento «LIBRO»
<!ELEMENT Autor (nombre, apellidos)> <!ELEMENT Editor (nombre, apellidos)>	Declara los elementos «Autor« y «Editor»
<!ELEMENT Título (#PCDATA)> <!ELEMENT nombre (#PCDATA)> <!ELEMENT apellidos (#PCDATA)> <!ELEMENT Precio (#PCDATA)>	Declara cada elemento hijo y el tipo de datos que pueden contener.

En el ejemplo anterior, se han utilizado una serie de símbolos para especificar las reglas de uso del elemento. Por ejemplo, el signo de interrogación indica un elemento de utilización opcional (puede estar ausente), el signo + indica que es requerido el elemento y puede aparecer tantas veces como sea necesario.

Que cada usuario pueda crear su propia DTD es una gran ventaja, ya que proporciona total libertad de adecuación a cada documento, pero también puede suponer un grave inconveniente, ya que es muy fácil que para documentos de un mismo sector (arquitectura, edición, educación, etc.), existan variadas DTD, haciendo muy difícil su manejo por usuarios distintos a los que hayan diseñado la información.

Por este motivo, en la actualidad se están definiendo DTD por grupos sectoriales con similares intereses, de forma que existirán DTD estándares avalados por organizaciones que garanticen que cualquier usuario que las adopte como suyas, trabaje con las mismas etiquetas e idénticas normativas (de forma similar al actual HTML). Como ejemplos de estas DTD estándares tenemos:

- CDF - *Channel Definition Format* (canales para envio de información periódica),
- CML - *Chemical Markup Language* (información del sector químico),
- MathML - *Mathematical Markup Language* (datos matemáticos),
- SMIL - *Synchronized Multimedia Integration Language* (presentaciones de recursos multimedia).

3.2 Esquemas

Últimamente se está imponiendo otra forma más eficaz de definición de elementos, conocida como «esquema». Los esquemas son una innovación reciente en XML, y seguramente suplantarán a las DTD en un futuro cercano. Se puede definir como una DTD que permite su ampliación mediante un lenguaje de definición de esquemas *(XML Schema)*. La funcionalidad de un esquema es equivalente a la de una DTD, pero está escrito en la sintaxis XML. Asegura que los documentos se adhieran a un vocabulario predeterminado, y permite extender la funcionalidad de una DTD con tipos de datos, herencia y reglas de presentación.

```
<?xml versión=«1.0»>
<schema>
    <element name=»CATALOGO»>
        <type>
            <element name=»LIBRO» type=»string»/>
                <type>
                    <attribute name=»ISBN» type=»string»>
                    <element name=»titulo» type=»string»>
                    <element name=»autor» type=»string»>
                    <element name=»editorial» type=»string»>
                    <element name=»precio» type=»fr:currency»>
                </type>
            </type>
        </element>
</schema>
```

3.3 Namespaces

Puesto que XML fue creado para permitir la interoperabilidad, y todo el mundo puede crear sus propios vocabularios XML, se produciría una terrible confusión si diferentes desarrolladores escogiesen los mismos nombres de elementos para representar diferentes entidades. Por este motivo fueron introducidos los «espacios de nombres» (*NameSpaces*) en XML para resolver este problema, permitiendo el uso de múltiples vocabularios en el mismo documento.

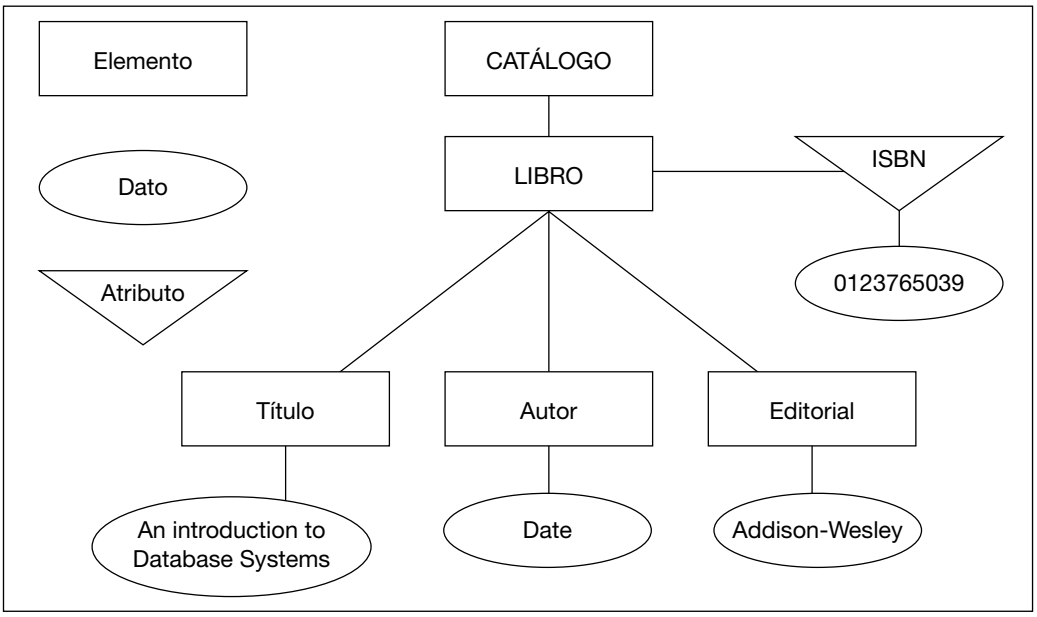
Un *namespace* es un vocabulario definido dentro de un URI (*Universal Resource Identifier, Identificador de Recursos Universal*). Los *namespaces* se usan para resolver conflictos de nombres entre elementos en un documento XML cuando los elementos se derivan de diferentes fuentes.

La siguiente declaración de *namespace* (xmlns) asocia el prefijo «edi» con el *namespace* <http://negocioe.org/esquema>. El nombre de elemento «precio» usa el prefijo para crear un nombre cualificado, indicando que el elemento precio se deriva del *namespace* especificado.

```
<x xmlns:edi=«http://negocioe.org/esquema»>
        <edi:precio unidades=«Euro»>30.15</edi:precio>
</x>
```

3.4 Análisis sintáctico y modelo de objeto documento (DOM)

Siguiendo con el proceso que se desarrolla en las aplicaciones XML, después de recoger la información de todos los documentos que definen los datos XML, se genera internamente una estructura que organiza a los elementos que describen las etiquetas en forma de árbol jerárquico. Un documento XML puede verse como una colección de elementos (*Information Items*) estructurados en forma de árbol (*Information Set*). En caso que se detecte algún error incompatible con las estrictas normas XML, se interrumpe el proceso generando un error.



Todo este proceso se puede realizar gracias al analizador sintáctico (*parser*), que en la mayoría de los casos se relaciona directamente con el estándar DOM (*Document Object Model*, modelo de objeto documento). DOM es la definición de un conjunto de clases cuyas instancias representan documentos XML, después de que éstos sean procesados. La definición de DOM está basada en la sintaxis de XML.

Las reglas mínimas que hay que cumplir para no ser rechazados por un analizador XML son:

- Sólo se permite un elemento raíz. Es imprescindible cumplir esta norma para que el *parser* pueda saber que el documento está completo. Es el equivalente a la etiqueta <HTML> de HTML.

- Hay que incluir etiquetas de inicio y final para todos los elementos. No se permiten casos de etiquetas «sueltas» como las <P> o <LI> de HTML.

- En caso de trabajar con etiquetas «vacías» (normalmente llevan atributos), hay que incluir la «barra» (/) antes del signo «mayor» (>). (marcas únicas del tipo <ETIQUETA/>). Serían los casos típicos de <HR> o
 de HTML.

- Hay que anidar las etiquetas correctamente. Las etiquetas anidadas deben situarse en el mismo orden de apertura que de cierre: <ETIQUETA1> … <ETIQUETA2> … <ETIQUETA3> … </ETIQUETA3> … </ETIQUETA2> … </ETIQUETA1>.

- Es obligatorio que los valores de los atributos vayan entre comillas. Las etiquetas con atributos deben marcar sus valores entre comillas: <ETIQUETA ATRIBUTO=»x»/>. En HTML suele ser optativo.

- XML distingue entre mayúsculas y minúsculas, al contrario de HTML, que no se preocupa de que sus etiquetas estén en mayúsculas, minúsculas o mezcladas.

Cuando se usa una DTD, únicamente son procesados los documentos «válidos». El analizador sintáctico valida el documento XML, comparándolo con la DTD a la que está asociado. Un documento XML «bien-formado» (*well-formed*), es aquel cuyo código sencillamente sigue las reglas sintácticas de XML. Sin embargo para ser válido, un documento XML también debe conformar con una DTD.

4. Estilo y presentación

Puesto que el objetivo de XML es separar contenido de presentación, cualquier elemento XML debe estar enlazado con una CSS para ser visualizado. CSS especifica el formato de presentación: fuentes, colores, espaciado, etc. La manera en que deben presentarse los documentos HTML era inicialmente codificada en el navegador web. Las CSS han sido usadas durante bastante tiempo tanto en XML como en HTML.

El lenguaje extensible de CSS (XSL, Extensible Stylesheet Language) usa la sintaxis de XML y proporciona un conjunto de herramientas de formato mucho más completo que las de CSS.

En los sistemas de documentos más abiertos, como los basados en SGML, no puede codificarse la información de presentación en el navegador debido a la variedad de tipos de documentos que se aceptan en el sistema. Por tanto, estos sistemas necesitan un mecanismo de hojas de estilo como DSSSL (*Document Style Semantics and Specification Language*, lenguaje de especificación y semántica de estilo de documentos) para proporcionar presentación e información de estilo. DSSSL es un estándar basado en SGML que regula las normas de presentación de documentos de marcas para el web.

Los navegadores XML aceptan tipos del documento diferentes y por tanto también necesitan un lenguaje de estilo. Pero se consideraba que el lenguaje de estilo de la comunidad SGML era demasiado complejo como para darle un uso extendido en el web. Además, la separación estricta entre la estructura y presentación de DSSSL no encaja en las aplicaciones web actuales basadas en HTML donde esta distinción está menos clara. Se ha desarrollado XSL (*Extensible Stylesheet Language*, lenguaje de hojas de estilo extensible) como un subconjunto de DSSSL, con medios adicionales para operar en un ambiente basado en HTML.

Por ejemplo, si se define una hoja de estilo CSS ligada con un archivo HTML con el siguiente código:

```
P {font-family:Verdana; font-size:12 pt}
TABLE {border:2; font-family:Arial; font-size:10 pt}
H3 {font-family:Comic Sans MS; font-size:12 pt; color:blue}
```

Se indica al navegador que presente los textos incluidos entre <P> y </P> con un tipo de letra Verdana de 12 puntos, las tablas con una fuente Arial 10 puntos y un ancho de 2 en los bordes, y los titulares <H3> con una letra Comic Sans MS de 12 puntos y color azul.

Utilizar CSS con XML es similar, con la excepción de que las etiquetas son diferentes a las de HTML. Un código como el siguiente:

```
AUTOR {display:block; font-family:Arial; font-size:small}
TITULO {display:block; font-size:x-large; text-align:center; color:#996699}
```

Sería perfectamente válido para que los datos de las etiquetas <AUTOR> y <TITULO> se presentasen según su descripción.

CSS es eficaz para describir formatos y presentaciones, pero no sirve para decidir qué tipos de datos deben ser mostrados y cuáles no. Esto es, CSS se utiliza con documentos XML en los casos en los que todo su contenido debe mostrarse sin mayor problema.

XSL además de especificar la presentación de los datos de un documento XML como CSS, también permite filtrar los datos de acuerdo a ciertas condiciones. Se parece un poco más a un lenguaje de programación: posibilita la ejecución de bucles, sentencias del tipo IF…THEN, selecciones por comparación, operaciones lógicas, ordenaciones de datos, utilización de plantillas, y otras cuestiones similares.

A continuación se muestra un sencillo ejemplo de XSL que permitiría mostrar (en un documento HTML) todos los contenidos de las etiquetas <TITULO> y <AUTOR> de un documento XML, mediante un bucle sin condiciones:

```
<xsl:template match=»/»>
    <HTML>
    <BODY>
    <xsl:for-each selec=«/CATALOGO/LIBRO»>
        Título:
        <xsl:value-of select=»titulo»/><BR/>
        Autor:
        <xsl:value-of select=»autor»/><BR/>
    </xsl:for-each>
    </BODY>
    </HTML>
</xsl:template>
```

5. Transformación e integración de datos

La información electrónica no se adapta adecuadamente a los modelos de datos tradicionales, relacionales u orientados a objetos. Muy diversas aplicaciones almacenan sus datos en formatos de datos sin ningún estándar, sistemas propietarios, documentos estructurados en HTML o SGML.

Uno de los grandes beneficios de XML es su capacidad de transformar el código desarrollado para una aplicación, haciéndolo corresponder con el código desarrollado para otra. XSL se puede usar para especificar la presentación de un documento, como las hojas de estilo CSS, pero las propiedades extendidas de XSL también permiten transformar la estructura de un documento XML. Para ello es posible utilizar XSLT (XSL Transformation).

Como ejemplo, supongamos que una aplicación B usa los mismos datos que hemos especificado en nuestra DTD de libros pero en otro formato. La siguiente transformación permite la conversión entre los dos formatos:

```
<LIBRO
Titulo=»El gran libro de XML»
Autor=»Pepe Pérez»>
<ISBN> 010200030X</ISBN>
</LIBRO>
```

```
<?xml version=»1.0» ?>
<xsl: transform
xmlns:xsl=»http://www.w3.org/XSL/Transform/1.0» indent-result=»yes»>
<xsl:template match=»LIBRO «>
<LIBRO>
        <TituloLibro><xsl:value-of
        select=»@Titulo»/></TituloLibro>
        <AutorLibro><xsl:value-of
        select=»@Autor»/></AutorLibro>
        <ISBN><xsl:value-of
        select=»ISBN»/></ISBN>
</LIBRO>
</xsl:template>
</xsl:transform>
```

```
<LIBRO>
<TituloLibro>
El gran libro de XML
</TituloTitulo>
<AutorLibro> Pepe Pérez
</AutorLibro>
<ISBN> 010200030X </ISBN>
</LIBRO>
```

Otra aplicación importante de XML es el intercambio de datos electrónicos (EDI) entre dos o más fuentes de datos en la web. Los datos electrónicos deben orientarse al procesamiento por ordenador. Por ejemplo, los ingenios de búsqueda podrían automáticamente integrar la información de fuentes relacionadas que publican sus datos en XML; las organizaciones podrían publicar datos sobre sus productos y servicios, y los clientes potenciales podrían comparar y procesar esta información automáticamente; los socios comerciales podrían intercambiar los datos operacionales entre sus sistemas de información. También se presentarían nuevas oportunidades a terceros integrando, transformando, limpiando, y agregando datos XML. En este contexto, estamos tomando un enfoque de base de datos de XML. Con-

sideramos que un documento XML es una base de datos y su DTD es el *esquema* de la base de datos.

Las aplicaciones EDI requieren herramientas que soporten las siguientes tareas:

- extracción de datos de grandes documentos XML,
- conversión de datos entre bases de datos relacionales u orientadas a objetos y datos XML,
- transformación de datos de una DTD a otra DTD diferente, y/o
- la integración de datos XML de múltiples fuentes.

La extracción, conversión, transformación e integración son problemas bien conocidos en el enfoque de bases de datos. Sus soluciones se basan en un lenguaje de consulta relacional (SQL, Structured Query Language) u orientado a objetos (OQL, Object Query Language). Para XML se han propuesto diferentes lenguajes de consulta: *XML-QL, Lore, YATL y XQL* [Fernández, Simeon, Wadler, 1999].

Una pregunta obligada es por qué no se adapta SQL u OQL a XML. La respuesta es que los datos XML son fundamentalmente diferentes a los datos de los enfoques tradicionales «relacionales» u «orientados a objetos», y por consiguiente, ni SQL ni OQL son apropiados para XML. La distinción importante entre los datos XML y datos en los modelos tradicionales es que XML no se estructura rígidamente. En los modelos relacionales y orientados a objetos, cada instancia de datos tiene un *esquema* separado e independiente de los datos. En XML, el esquema existe con los datos como nombres de la etiqueta. Por ejemplo, en el modelo relacional, un esquema podría definir la relación «persona» con los atributos nombre y dirección, es decir, persona (nombre, dirección). Una instancia de este esquema contendría tuplas como («Pepe Pérez», «Valladolid»). La relación y nombres de atributos están separados de los datos y normalmente se almacenan en un catálogo de la base de datos (Beeri, Milo, 1999).

En XML, como ya hemos comentado, la información del esquema se almacena con los datos. Los valores estructurados se llaman *elementos.* Los atributos, o nombres de elementos, se denominan *etiquetas*, y los elementos también pueden tener *atributos* cuyos valores siempre son atómicos. Por ejemplo, <persona> <name>Pepe Pérez</name> <dirección>Valladolid </dirección> </persona> estaría bien formado en XML. De esta forma, los datos XML son autodescriptivos y pueden modelar irregularidades que no pueden planearse en los datos de modelos tradicionales.

Por ejemplo, los elementos de los datos pueden carecer de ciertos elementos o tener ocurrencias múltiples del mismo elemento; los elementos pueden tener valores atómicos en algunos elementos de datos y pueden estructurarse en otros; las colecciones de elementos pueden tener estructura heterogénea y la información relacionada semánticamente puede representarse de forma diferente en los elementos. Los datos con estas características se han denominado datos semiestructurados (Suciu, 1998). Incluso los datos XML con una DTD asociada se autodescriben (el esquema siempre se almacena con los datos) y, salvo formas restrictivas de DTD, pueden presentar todas las irregularidades descritas anteriormente. Esta flexibilidad es crucial para las aplicaciones de EDI.

Los investigadores de la comunidad de bases de datos han encontrado en este tipo de datos autodescriptivos un nuevo campo de trabajo. Estos datos son fundamentalmente diferentes de los datos de los modelos tradicionales. Los datos semiestructurados están siendo considerados en los problemas de integración de fuentes de datos heterogéneas y en el

modelado de fuentes como las bases de datos biológicos, la información web y documentos de texto estructurado, como SGML y XML. La investigación se ha dirigido hacia modelos de datos, lenguajes de consulta, procesamiento y optimización de consultas, y lenguajes de definición y extracción de esquemas (Skogan,1999).

Una observación importante es que los datos XML pueden considerarse un caso particular de datos semiestructurados. La información expresada en XML es notablemente similar a los datos semiestructurados: consiste en objetos y atributos (elementos y etiquetas respectivamente), y, como los datos semiestructurados, tiene el esquema incorporado en los mismos datos.

Como ejemplo de lenguaje de consulta para XML, consideremos XML-QL. Algunas de las características de XML-QL son las siguientes:

- Es declaratorio.
- Es «relacionalmente completo»; en particular, puede expresar combinaciones (*joins*).
- Es lo bastante sencillo como para que técnicas de bases de datos conocidas como optimización de consultas, estimación de costes, etc., puedan extenderse a XML-QL.
- Puede extraer datos de documentos XML existentes *y* construir nuevos documentos XML (Deutsch y otros, 1999).

Como ejemplo de XML-QL consideremos el siguiente documento XML sobre libros. La consulta recupera el título y autor de libros publicados por «Addison-Wesley» y los agrupa en un nuevo elemento <result>:

Documento origen

```
<CATALOGO>
<LIBRO isbn=»0123765039»>
   <titulo>An Introduction to
   Database Systems </titulo>
   <autor>Date </autor>
   <editorial>Addison-Wesley
   </editorial>
</LIBRO>
<LIBRO isbn=»076599825X»>
   <titulo>Foundation for Object
   Databases</titulo>
   <autor>Date </author>
   <autor>Darwen </autor>
   <editorial>Addison-Wesley
   </editorial>
 </LIBRO>
</CATALOGO>
```

Consulta

```
WHERE <LIBRO>
   <editorial>Addison-Wesley</>
   <titulo> $t </>
   <autor> $a </>
</> IN «www.a.b.c/libros.xml»
CONSTRUCT <result>
   <autor> $a </>
   <titulo> $t </>
</>
```

Resultado

```
<result>
      <autor>Date </autor>
      <titulo>An Introduction to
      Database Systems </titulo>
</result>
<result>
      <autor>Date </autor>
      <titulo>Foundation for
      Object Databases</titulo>
</result>
</result>
      <autor>Darwen </autor>
      <title>Foundation for Object
   Databases</titulo>
</result>
```

Para una aplicación hipermedia cuyos datos provienen de un conjunto de aplicaciones distinto, los datos pueden estar almacenados en un SGBD relacional, documentos XML, etc. Los datos tendrán una estructura y origen heterogéneo. Al trabajar con distintas fuentes de datos es aplicable la arquitectura de los sistemas federados de bases de datos (SFBD), donde se distinguen los siguientes esquemas:

- **Esquema local.** Datos y estructura tal y como los facilita la aplicación emisora. En el caso de un *SGBD* relacional, el esquema será un conjunto de tablas interrelacionadas. Si la aplicación ya facilita XML, el esquema local corresponderá a la DTD asociada. Finalmente si se trata de un fichero de texto, el esquema local será la estructura del registro.

- **Esquema componente.** Representación del esquema local en términos del modelo canónico de datos. En nuestro caso, escogemos XML como lenguaje común. Si la aplicación emisora no utiliza este modelo, será necesario un conversor para pasar el esquema local a XML.

- **Esquema federado.** Esquema resultante de la integración de varios esquemas exportados. Para ello, se utilizan mecanismos de vistas. Para el caso de documentos XML, estos mecanismos todavía están en sus principios. Una posibilidad es utilizar XSLT o XML-QL (Díaz, Iturrioz e Ibáñez, 2000).

6. Navegación

Una de las características de todo documento web es la inclusión de enlaces de todo tipo: a imágenes, a sonidos, a vídeos, a otros párrafos, a otros documentos, etc. En HTML la cuestión está resuelta, ya que existen etiquetas para cada caso. Pero la solución no satisface a todos los autores.

Mientras la comunidad de SGML criticó a HTML por sus pobres mecanismos de marcado, la comunidad del hipertexto criticó el limitado soporte de HTML a la definición de hiperenlaces. Los enlaces simples de HTML no pueden competir con los tipos de enlaces más avanzados de muchos de los sistemas de hipertexto tradicionales. La vinculación simplista que el web plantea era especialmente problemática para construir sistemas de hipertexto abiertos: soportar los tipos de enlaces de las aplicaciones existentes (procesadores de texto, hojas de cálculo, etc.) e integrar los sistemas de hipertexto más poderosos en el ambiente web.

Sin embargo, ha sido reconocida por la comunidad SGML la necesidad de que el marcado soporte los requisitos de enlaces más complejos. Tanto el estándar HyTime como TEI (Text Encoding Iniciative) han desarrollado convenciones comunes para la codificación de estructuras avanzadas de enlaces. Pero de nuevo, estas soluciones nunca alcanzaron una gran difusión en la web, debido a la complejidad de las normas. Así que el hueco existente entre el marcado sencillo pero limitado de HTML y el marcado avanzado pero más complejo de SGML es bastante similar al hueco que existe entre los medios de vinculación sencillos y limitados de HTML comparados con el soporte a la vinculación avanzado, pero complejo de HyTime y TEI. La cuestión de los enlaces e hipervínculos es tan importante para los documentos XML que el W3C ha sacado las especificaciones que las controlan fuera de las descripciones de DTD, creando dos normas: XLink y Xpointer; que definen un conjunto de estructuras de enlaces y los métodos de direccionamiento, combinando lo mejor de los enlaces HTML, HyTime y TEI.

XLink (anteriormente conocido como XLL, *Extensible Linking Language*) define la forma en la que los documentos XML deben conectarse entre sí (determina el documento al que se desea acceder). XPointer describe cómo se puede apuntar a un lugar específico de un determinado documento XML.

Las especificaciones de los hipervínculos para XML proporcionan muchas más posibilidades que las de la etiqueta <A> de HTML: adherirse a cualquier etiqueta, hacer referencia a un lugar concreto de un documento determinado a través de su nombre o localización, descripción de hipervínculos en documentos externos, distintas formas de procesamiento, multifuncionalidad (permitir varios saltos), etc.

Al contrario de lo que ocurre con HTML, en XML existen dos tipos básicos de hipervínculos: simples y extendidos.

Un ejemplo de hipervínculo simple sería:

```
<AUTOR xlink:href=»autores.xml#Pepe» xlink:show=»new»>
<NOMBRE>Pepe</NOMBRE>
</AUTOR>
```

Otro ejemplo de hipervínculo extendido podría ser:

```
<EDITOR_AUTOR xlink:extended>
<xlink:locator href=»#Goldfarb « id=»editor»/>
<xlink:locator href=»autores.xml#Date» id=»autor»/>
<xlink:arc from=»editor» to=»autor» show=»replace»/>
</EDITOR_AUTOR xlink:extended>
```

En el primero se puede observar la definición de un hipervínculo simple que se abre en una nueva ventana (*show=»new»*), mientras que en el segundo se define un hipervínculo con tres posibilidades diferentes: a una sección determinada del documento (*#Goldfarb*), o a un determinado lugar de otro documento (*autores.xml#Date*), o a una zona delimitada por dos marcadores (*editor* y *autor*). Aunque no se han comentado todas las posibilidades de los hipervínculos XML, sí debe quedar claro que los enlaces XML son más variados que los que nos proporciona la sencilla y conocida etiqueta <A> del HTML.

7. Conclusiones

Desde la aparición de la norma XML 1.0 en 1998 hemos asistido a un inusitado crecimiento de una tecnología prometedora. XML ha puesto a disposición de un gran número de desarrolladores la potencia del lenguaje generalizado de marcas. Ha sido considerado como el «SGML para las masas». Su éxito está precisamente en la proliferación de lenguajes basados en él. Podemos encontrar innumerables aplicaciones de XML en casi todas las áreas.

Para evitar la aparición de innumerables formas de describir documentos de un determinado sector, ya se está trabajando en la definición de DTD sectoriales de carácter público que estén arropados por el máximo de empresas y organismos posibles. Según se vayan publicando, se crearán nuevas herramientas para su tratamiento.

La fuerte utilización de XML es un signo de su fuerza, claridad y simplicidad. Bastantes de los estándares de la familia XML tienen una fuerte razón de ser, son concisos y útiles, como las normas centrales de XML comentadas *Namespaces, XSLT y Xlink.* Aunque la versión 1.0 de XML es ya definitiva, no pasa lo mismo con las demás normas que le acompañan, que poco a poco van pasando de «borrador de trabajo» a «recomendación». Aunque todos los grandes de la informática como IBM, Sun**,** Oracle y Microsoft tienen fuertes apuestas en XML, en general, a las empresas les cuesta mucho invertir en desarrollo de productos que no estén soportados por estándares definitivos.

La popularización llegará cuando todas las herramientas relacionadas con Internet sean capaces de trabajar con XML. Existen varios navegadores que admiten XML, como Explorer, Amaya o HotMetal y casi cada día están saliendo nuevos productos (analizadores, visualizadores, ingenios, editores, DTD, etc.). La versión 4.x de Netscape Navigator no soporta XML, y aunque se supone que la versión 5 sí lo permitirá, la política comercial actual de Netscape no permite asegurar cuándo ni cómo será.La existencia de XML no implicará la desaparición del HTML. La mayoría de las páginas web actuales son documentos sencillos de texto con algunas imágenes, y HTML seguirá siendo el medio más eficaz para crearlas y publicarlas, aunque probablemente se tenderá a utilizar la versión XHTML.

Si la evolución continúa como hasta este momento, XML se convertirá en el lenguaje que garantizará el intercambio de cualquier información y transformará la web en un sistema hipermedia abierto, sin problemas de separación entre contenido y presentación.

Bibliografía

BEERI, C.; MILO, T. (1999). «Schemas for Integration and Translation of Structured and Semi-Structured Data.» Proceedings of the International Conference on Database Theory, Jerusalem, Israel,1999. Springer Verlag. <ftp://ftp.math.tau.ac.il/pub/milo/icdt99-2.ps.Z>, 1999.

BRAY, T.; HOLLANDER, D.; LAYMAN, A. (eds.). (1999) «Namespaces in XML.» World Wide web Consortium. W3C Recommendation. <http://www.w3.org/TR/REC-xml-names>, enero, 1999.

BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C.M. (eds.). (1998). «Extensible Markup Language (XML) 1.0». World Wide web Consortium. W3C Recommendation. <http://www.w3.org/TR/1998/REC-xml>, febrero, 1998.

CLARK, J. (ed.). (1999). «XSL Transformations (XSLT) V1.0.» World Wide web Consortium. W3C Recommendation. <http://www.w3.org/TR/1999/REC-xslt-19991116>, noviembre, 1999.

DANIEL, R.; DEROSE, S.; MALER, E. (eds.).(1998). «XML Pointer Language (XPointer) V1.0». World Wide web Consortium, 1998. <http://www.w3.org/TR/xptr>, 1998.

DEROSE, S.; MALER, E.; ORCHARD, D.; TRAFFORD, B. (eds.). (2000). «XML Linking Language (XLink) V1.0». World Wide web Consortium. Candidate Recommendation 3 July 2000. <http://www.w3.org/TR/2000/CR-xlink-20000703>, julio, 2000.

DEUTSCH, A.; FERNÁNDEZ, M.; FLORESCU, D.; LEVY, A.; SUCIU, D. (1999). A query language for XML. International World Wide web Conference. http://www.research.att.com/~mff/files/final.html>.

DÍAZ, O.; ITURRIOZ, J.; IBÁÑEZ, F. (2000) «Integración, navegación y presentación: experiencias utilizando XML.» *NOVATICA*, n. 146, p. 12-19. julio-agosto, 2000.

FERNÁNDEZ, M.; SIMEON, J.; WADLER, P. (1999). «XML Query Languages: Experiences and Exemplars.» <http://www.w3.org/1999/09/ql/docs/xquery.html>, 1999.

MONTERO, R. (2000). «Guía XML: Evolución». <http://www.ramon.org/xml/guiaXML/guiaXML01.htm>, marzo, 2000.

VAN OSSENBRUGGEN, J.; ELIËNS, A.; RUTLEDGE, L. (1998) «The Role of XML in Open Hypermedia Systems.» 4th Workshop on Open Hypermedia Systems, Hypertext '98. Pittsburgh, June 20-24, 1998. <http://www.daimi.au.dk/~kock/OHS-HT98/Papers/ossenbruggen.html>, 1998.

SKOGAN, D. (1999). «UML as a Schema Language for XML based Data Interchange.» Unified Model Language Conference UML'99. <http://www.cs.colostate.edu/UML99/>, 1999.

SUCIU, D. (1998). «An overview of semistructured data». SIGACT News, 29(4), 28-38, diciembre. <http://www.research.att.com/~suciu/strudel/external/files/_F242554565.ps>.

W3C HTML working group. «XHTML 1.0: The Extensible HyperText Markup Language.» World Wide web Consortium. W3C Proposed Recommendation. <http://www.w3.org/TR/xhtml1>, enero, 2000.