Artículos

Sgml y servicios de información

Por **Eduardo Peis y Félix de Moya**

Eduardo Peis

Resumen: *No es concebible la gestión "actual" de recursos informativos sin la web. Para el tratamiento, conservación, difusión y acceso de la información electrónica en la Red es necesario disponer de una norma que asegure la independencia de fabricantes y permita extraer la mayor cantidad de posibilidades hipertexto e hipermedia de la web. Este trabajo presenta la norma sgml y su "familia" de adaptaciones (xml, xsl, xlink) como potencialmente ideales en su aplicación a los servicios de información.*

Palabras clave: *Información electrónica, Metalenguajes de etiquetado, Sgml, Servicios de información, Xml.*

Félix de Moya

Title: Sgml and information services

Abstract: *Present management of information resources is inconceivable without the web. For the handling, preservation, dissemination and access to electronic information on the Net, a standard is needed to ensure independence from manufacturers and to take full advantage of the hypertext and hypermedia capabilities of the web. This paper presents the sgml standard and its "family" of variations (xml, xsl, xlink) as potentially ideal for information service applications.*

Keywords: *Electronic information, Markup meta-languages, Sgml, Information services, Xml.*

Peis, **Eduardo; Moya**, **Félix de.** *"Sgml y servicios de información". En: El profesional de la información, 2000, junio, v. 9, n. 6, pp. 4- 17.*

La norma sgml

Sgml (*standard generalized markup language)* comenzó a desarrollarse en *IBM* en 1969, denominándose en un primer momento GML (*generalized markup language*), que, anecdóticamente, coincidía con las iniciales de los apellidos de sus tres primeros diseñadores (**Goldfarb**; **Mosher**; **Lorie**). GML fue el resultado de un esfuerzo por encontrar una vía para conformar la documentación básica de la empresa en un formato electrónico transferible y gestionable (**Cole**; **Kazmer**, 1995). La idea fue ampliada hasta llegar a convertirse en la norma sgml durante una reunión del *Ansi* (*American National Standards Institute*) en 1978. Fue considerada como una norma por la *ISO* en 1986, adoptada como *ISO 8879*, formando parte de un conjunto más amplio bajo el título *Information processing-text and office systems* (ver *ISO-sgml* en la bibliografía). Una vez declarada norma internacional, **Goldfarb** (1990) se reincorporó a los trabajos para su desarrollo (**Adler**, 1992). Sgml posibilita la estructuración lógica de documentos electrónicos para ser representados explícita y rigurosamente de forma inequívoca e independiente de aplicaciones y sistemas (**Corthouts**; **Philips**, 1996).

Por *markup language* (lenguaje de etiquetado) se designa a un conjunto de convenciones de marcaje que se usan juntas para codificar textos. Un lenguaje de esta naturaleza debe especificar: 1) qué marcas son admitidas, 2) cuáles son indispensables, 3) cómo se distinguen las marcas del texto y 4) qué significa cada una de éstas. Sgml proporciona los medios para llevar a cabo las tres primeras especificaciones; para la última se necesita concretar una serie de directrices (**Van Herwijnen**, 1994).

Hay tres características de sgml que lo distinguen de otras normas para lenguajes de etiquetado:

—Énfasis sobre el etiquetado descriptivo más que en el de procedimiento,

—concepto de *document type* (tipo de documento), e

—independencia de cualquier sistema de representación del alfabeto en que esté escrito el texto (**Marcoux**; **Sévigny**, 1997).

Por supuesto, la idea de etiquetar un texto no es nueva. Hasta muy recientemente los maquetadores de las imprentas utilizaban códigos especiales para indicar al cajista el diseño que debía aplicar: tamaño de

Original recibido el 4-2-00
Aceptación definitiva: 7-4-00

fuente, centrados, resaltados, etc. Los modernos procesadores de texto utilizan códigos "invisibles" al usuario que determinan tamaño y tipo de letra, cambios de línea, tabulaciones, etc. En ambos casos son códigos de etiquetado que se pueden considerar como "de procedimiento". Sgml, por el contrario, como sistema de marcado descriptivo usa marcas que proporcionan nombres para categorizar partes de un documento consiguiendo así, entre otras muchas funcionalidades, separar el documento de su formato o que el documento pueda ser procesado fácilmente con muchos tipos diferentes de software, cada uno de los cuales puede aplicar diferentes instrucciones de procesamiento (**Sperberg-McQueen**; **Burnard**, 1994).

Sgml introduce la noción de *document type* y, consiguientemente, una *document type definition —DTD* (definición del tipo de documento)—. Como su propio nombre indica, ésta se emplea para definir un tipo de documento de acuerdo con sus partes constituyentes y la estructura lógica que adoptan dichas partes.

Sgml proporciona además un instrumento de aplicación general para la *string substitution* (cadena de sustitución) que es una forma simple, independiente de mecanismos, de concretar que un grupo de caracteres específico del documento deberá ser reemplazado por algún otro cuando sea procesado. Una aplicación obvia es asegurar la consistencia de nomenclatura; otra, aún más importante, es evitar la incapacidad que muestran diferentes sistemas informáticos para aceptar el conjunto de caracteres de los demás o la imposibilidad de algún modelo para ofrecer todos los caracteres gráficos que necesita una aplicación concreta, proporcionando representaciones descriptivas para caracteres no transferibles. Aquellos conjuntos definidos mediante este mecanismo de cadena de sustitución son llamadas *entities* (entes o entidades) (**Goldfarb**, 1997).

Familia de normas sgml/xml

Se emplea el término *familia* para indicar que, desde que fuera aprobada como norma internacional en 1986, su uso se ha extendido y, alrededor de la considerada como *matriz*, se ha desarrollado una serie de normas y/o especificaciones subsidiarias y/o complementarias que constituyen un *grupo* normativo que sigue creciendo.

Además de las estrechamente ligadas al desarrollo de sgml, como la *ISO 10179-1996 document style semantics and specification language* (*ISO-dsssl*), que es también la base de otros sistemas de definición de estilos como c*ascading style sheets* (ver *W3C-css2*), el aplicado en html (**Adler**, 1997); o *HyTime*, *ISO 10744-1994 hypermedia/time-based structuring language* (ver *ISO-HyTime*), que puede ser considerado como una extensión de sgml, y que proporciona normas para establecer enlaces hipertexto en documentos sgml (**Baird** et al., 1994). Existe un grupo de especificaciones y recomendaciones derivadas de sgml que forman un conjunto de aplicaciones normativas que se complementan unas a otras. De hecho, en la literatura generalista más reciente se habla de sgml/xml como conceptos inseparables (*sgml/xml webpage*).

En términos generales, la preocupación fundamental de los servicios de información debe ser diseñar e implementar sistemas eficaces para procesar y difundir la denominada información documental. La rápida evolución de estos servicios en los últimos años ha venido de la mano de las llamadas nuevas tecnologías (para producir, procesar y difundir información documental). Si en algo se ha modificado dicha información es que ahora un porcentaje muy significativo de la misma es electrónica (**Marcoux**; **Sévigny**, 1997).

«Sgml usa marcas que proporcionan nombres para categorizar partes de un documento consiguiendo así, entre otras muchas funcionalidades, separar el documento de su formato»

No es concebible la actual gestión de recursos informativos sin la web. En este escenario es donde el concurso de la *familia* de normas sgml puede resultar clave: asegura la consistencia (en la producción, el procesamiento, el almacenamiento y la distribución), proporciona una enorme flexibilidad (en la presentación y en el formato, por ejemplo), incrementa la discriminación informativa, lo que repercute en una mejora del control y la recuperación de información, etc.

Si de lo que se trata es de explotar las potencialidades hipertexto e hipermedia de la Red, se puede advertir que html, con sus previsiblemente continuas actualizaciones, no es adecuado (**Burnard**, 1997). En efecto, es un lenguaje de anotación genérico, y además híbrido, que especifica muy poco sobre la estructura del documento, combina tipografía y contenido al mismo tiempo, está ligado a una serie de significados y no tiene una constitución arbitraria. Sgml, por su parte, sí proporciona estructura arbitraria, pero es demasiado difícil de implantar para un buscador en la web, contiene muchas características funcionales que raramente son usadas y que complican el desarrollo de software específico y, sobre todo, es prácticamente imposible procesar un documento sgml sin disponer de la *DTD* empleada para desarrollarlo (**Weibel**, 1995). Sgml hace que las posibilidades se incrementen enormemente con respecto a html, pero es precisamente por esto por lo que su implementación en la Red resulta cuando menos costosa (por no decir impracticable).

En realidad, html es una *DTD* sgml. Los continuos intentos para su mejora han desembocado en sucesivas versiones pero, en la práctica, este desarrollo no puede ser más que añadidos a la *DTD* original. Más fácil, y práctico, era reducir de alguna manera el conjunto sgml. Bajo los auspicios del *W3C* (*World Wide Web Consortium*), en 1996 se iniciaron los trabajos para la adaptación, y el resultado, bajo responsabilidad del *W3C's xml working group*, fue xml (*extensible markup language*), un subconjunto (o simplificación) adaptado de sgml que tiene la intención de aprovechar sus máximas ventajas posibles, pero permitiendo su implementación en la Red. Xml y sus especificaciones/recomendaciones complementarias se presentan, por tanto, como herramientas eficaces para la distribución y/o gestión de documentos electrónicos en la Red (**DeRose**, 1997).

Estrictamente hablando, xml es también un metalenguaje de etiquetado que será definido (cuando sean definitivamente aprobadas) mediante cuatro especificaciones: xml, propiamente dicho, *xlink* (*extensible linking language*), *xsl* (*extensible stylesheet language*) y *xua* (*xml user agent*), que aún no ha tomado siquiera la forma de borrador. No obstante, son muchos más los desarrollos relacionados con xml, como por ejemplo *xsql* (*extended structured query language*) o *sox* (*schema for object-oriented xml*), aunque la mayoría tienen estatus de "recomendación al *W3C*".

Los servicios de información y sgml/xml

Hay abundantes proyectos que se han desarrollado en el ámbito de la información y que cuentan con sgml como un componente esencial, la gran mayoría de ellos con la participación no sólo de servicios de información, sino también de grandes editores, comunidades académicas y organismos oficiales. Por su importancia como esquemas sgml para la descripción/codificación de recursos informativos, es necesario hacer referencia a los modelos *Text encoding initiative* (*TEI*) y *Encoded archival description* (*EAD*).

TEI

Comenzó siendo un proyecto de investigación internacional patrocinado por la *Association for Computing in the Humanities* (*ACH*), la *Association for Literary and Linguistic Computing* (*Allc*) y la *Association for Computational Linguistics* (*ACL*). *TEI* ha desarrollado esquemas de codificación modular sgml para una amplia variedad de tipos de documentos. Como consecuencia, el número de textos electrónicos en el área de humanidades ha crecido considerablemente y muchas bibliotecas e instituciones universitarias han comenzado a ampliar sus colecciones de textos electrónicos basadas en estas directrices. Las mejor conocidas son la *Oxford Text Archive*, el *University of Virginia Text Center*, el *Humanities Text Initiative* (*Universidad de Michigan*), el *British National Corpus Project*, etc. (**Hockey**, 1994).

El esquema *TEI* presenta algunas ventajas con relación a las particularidades de la metainformación necesaria para la documentación de recursos informativos electrónicos (**Palowitch**; **Horowitz**, 1996):

—Control documental: un texto *TEI* se compone de un encabezamiento *TEI*, seguido del texto en sí. El encabezamiento consta, a su vez, de cuatro elementos que proporcionan información (principalmente bibliográfica) relativa al documento como fichero electrónico.

—Flexibilidad: existe un gran número de *DTD*s *TEI*, con gran variabilidad en cuanto al uso de sus elementos. Las directrices *TEI* (**Sperberg-McQueen; Burnard**, 1994) son herramientas muy útiles en este sentido, ya que proporcionan recomendaciones sobre qué características codificar y cómo hacerlo.

«Muchos proyectos de bibliotecas digitales utilizan sgml: como formato de intercambio, para facilitar la indización con documentos de formatos dispares, para mejorar y acelerar el proceso de recuperación, etc.»

El encabezamiento *TEI* constituye una de las principales contribuciones para la codificación sgml de información bibliográfica (**Giordano**, 1994). Las directrices *TEI* incluyen una sección especial (24.3) sobre los elementos de este encabezamiento y sus relaciones con los registros *Marc*. Estos elementos contienen metainformación sobre el texto en sí, con la particularidad de que algunas partes de estos encabezamientos han sido realizados siguiendo fielmente normas de descripción existentes en los catálogos bibliotecarios como *Aacr2*, *Isbd*(*G*) y *Usmarc* (**Giordano,** 1994). Esto no es sorprendente, sabiendo que dicho encabezamiento fue diseñado por el *Committee on text documentation* compuesto, entre otros, por archiveros y bibliotecarios.

Dunlop (1995) comenta el uso del *TEI header* en el proyecto *British National Corpus Project* para la descripción a nivel de colección. Esta información descriptiva, que es común para muchos textos de este repertorio, puede ser utilizada para la generación automática de encabezamientos en una estructura de base de datos relacional.

El proyecto *Digital imaging* de la *Bodleian Library*, de Oxford, utiliza el modelo *TEI* para la creación de la base de datos como un esquema de metadatos conectados a las imágenes digitalizadas. Los resul-

tados de recuperación proporcionan una imagen del documento con sus datos bibliográficos asociados (**Gartner**, 1997).

EAD

Comenzó a desarrollarse en el marco del *Berkeley finding aids project* (*Bfap*), empleando sgml como técnica ideal para aplicar la codificación normalizada a los instrumentos de descripción archivística. El resultado de este proyecto fue una *DTD* que define una clase de documentos que, en términos generales, constan de una página de título opcional, la descripción de una unidad de material archivística y unos apéndices también opcionales (**Pitti**, 1999).

En la actualidad, el uso de *EAD*, bajo mantenimiento de la *Library of Congress* de Estados Unidos y responsabilidad de la *Society of American Archivists*, se ha generalizado para las descripciones de documentos de archivo (potencialmente conectadas a sus correspondientes imágenes digitales) y su difusión a través de la Red. *Online Archive of California* (*OAC*), *Harvard/Radcliffe digital finding aids project* (*Dfap*), *Yale finding aids project* (*Yfap*) o *American heritage project* (*AHP*) son algunos de los principales proyectos de aplicación *EAD*.

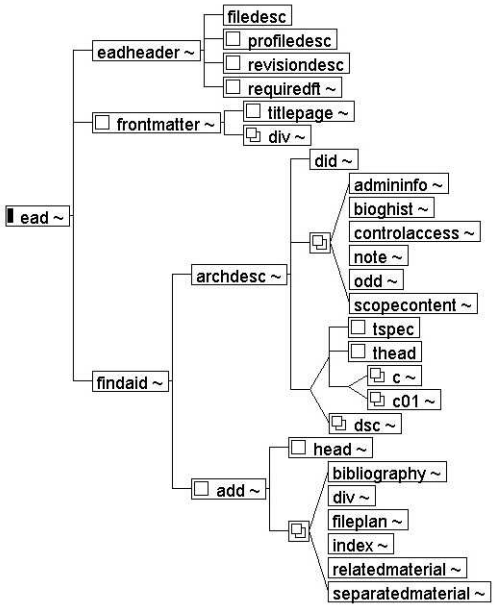
La *DTD* —actualmente en su versión 1.0, que posibilita el empleo de xml— (*SAA*, 1999), ha sido diseñada para reflejar la jerarquía natural que presenta la organización de los fondos, en conjunción con el orden intelectual que imponen los archiveros con sus prácticas descriptivas. Contiene dos tipos de elementos: los que codifican puntos específicos en la definición de partes componentes del instrumento de descripción o el material que describe (descriptivos); y los que podrían codificar cualquier característica del documento (genéricos). Estos últimos normalmente están incluidos en los primeros. A un nivel muy básico, un modelo de documento "instrumento de descripción" codificado utilizando *EAD*, consta de tres segmentos:

—Uno que proporciona información sobre el instrumento de descripción en sí mismo: su título, compilador, fecha de compilación (<*eadheader*>, que está basado en el *TEI header*);

—un segundo componente que incluye las cuestiones preliminares necesarias para la publicación formal del instrumento de descripción (<*frontmatter*>), y

—un tercero que proporciona la descripción del material archivístico en sí misma, además de la información contextual y administrativa asociada (<*findaid*>).

En una detallada comparación de tres formatos de metadatos sgml —*TEI*, *EAD* y *Cimi* (*Computer interchange of museum information*)—, **Burnard** y **Light**



Estructura de los principales elementos EAD

(1996) concluyeron que los componentes del encabezamiento *EAD* difieren ligeramente de los del *TEI*. Estos autores consideran la *EAD* como metadatos puros, dado que los instrumentos de descripción en sí mismos son metadatos. En este sentido, el *EAD header*, que describe el instrumento de descripción, puede ser considerado como meta-metadatos.

Como se decía, es posible comentar otras muchas investigaciones. Por ejemplo, el proyecto sgml de la *Vubis-Antwerpen Library Network* de la *Universidad de Amberes* (Bélgica) incluye el desarrollo de aplicaciones sgml locales (intercambio de información bibliográfica local, gateways para el préstamo interbibliotecario, servicios de documentación bibliotecaria en la web, actualización continua, etc.); también cuenta con aplicaciones mixtas (integración de información sgml externa con los servicios locales, por ejemplo para agilizar el proceso de acceso a documentos electrónicos) y con otras a gran escala (colecciones de artículos a texto completo de revistas electrónicas) (**Corthouts**; **Philips**, 1996).

HyperLib (**Philips**; **Corthouts,** 1995) fue un proyecto financiado por la *Comisión Europea*, el *Instituto de Investigación Hsat* (*Loughborough University,* UK) y la *Universidad de Amberes*, centrado principalmente en el desarrollo de guías hipertexto (para los usuarios finales de la biblioteca) y manuales (para el personal del centro) y recursos relativos a la base de datos (una bibliografía universitaria y un gateway web al catálogo) (HyperLib, 2000).

Elsa (*Electronic library sgml applications)* es un proyecto marco, también financiado por la *CE* (*Libraries programme*, hoy integrado en *Information Society Technologies*) cuyo objetivo es proporcionar un sistema de distribución de documentos electrónicos a las

bibliotecas y a los usuarios finales basado en sgml (**Adam**, 1995). En el *Consorcio Elsa* están implicados *Elsevier Science* (Holanda), encargado de la distribución de documentos codificados en sgml; *Jouve* (Francia), responsable de la creación del entorno servidor y del desarrollo de los mecanismos para el control de acceso y gestión; y *De Montfort University* (UK), que ha desarrollado la aplicación cliente. Los artículos son recibidos directamente de los editores en sgml. Algunos elementos definidos en la *DTD* son seleccionados para la creación de índices, y los documentos en sí mismos son transformados para su manipulación, transmisión en línea o navegación hipertexto. Los usuarios pueden diseñar perfiles DSI y recibir información sobre nuevos documentos cada semana.

«Mantener una o varias bases de datos sgml y filtrarlas a xml para las operaciones de indización y distribución en la web se conforma como una estrategia a largo plazo con amplias posibilidades de éxito»

Decomate (*Delivery of copyright materials to end-users)* es también un proyecto financiado por la *Comunidad Europea* y desarrollado en cooperación entre la *Universidad de Tilburg* (Holanda), la *London School of Economics* (UK) y la *Universitat Autònoma de Barcelona*. El objetivo es proporcionar a los usuarios finales acceso, a través de la biblioteca, a materiales con derechos de autor distribuidos por editores en formato electrónico. El sistema les permite buscar algún elemento informativo en un sistema bibliográfico (mediante una interfaz web que se ajusta a Z39.50), recuperar elementos de la base de datos y solicitar documentos (imágenes) para que aparezcan en pantalla o sean distribuidos de forma impresa.

La totalidad de la información de la base de datos del proyecto *Cheshire II* se almacenó etiquetada como un documento sgml. Mediante el uso de etiquetas para toda la información de la base de datos y la adopción de una *DTD* con el fin de definir la estructura de cada fichero de información, el sistema dispone de un formato común para diversos tipos de información que van desde documentos a texto completo y registros bibliográficos estructurados hasta otros hipertexto y multimedia complejos. Sgml también se usa como formato básico de los ficheros de configuración de *Cheshire II*, los cuales definen los elementos físicos de la base de datos del modelo, incluyendo la localización de los ficheros de datos, qué *DTD* sgml describe el fichero y la información sobre la que se crean los índices y los elementos que deberían contener (**Larson** et al., 1996).

El concurso de la familia de normas sgml en el contexto de las denominadas bibliotecas digitales ha sido, y es, importante. *Adonis*, *Tulip*, *Mercury* o el proyecto *Kwik*, de la *Universidad de Tilburg* son iniciativas que han empleado sgml bien conocidas en este área. De hecho, muchos proyectos de bibliotecas digitales utilizan sgml (Alexandria, Berkeley, Illinois, Standford, etc.): como formato de intercambio para facilitar la indización de documentos con formatos dispares, para mejorar y acelerar el proceso de recuperación, etc.

El proyecto de biblioteca digital *Envision* (**Heath** et al., 1995), por ejemplo, utiliza codificación sgml para las entradas a texto completo y bibliográficas. *Core* (**Entlich**, 1995) es un proyecto que recogió cinco años de las principales revistas sobre química, publicadas por la *American Chemical Society,* en formato electrónico. Para cada página de la base de datos de la editora, *Core* incluye tanto su imagen escaneada como su versión ascii etiquetada en sgml.

Son aún más numerosos los proyectos xml que se están emprendiendo. Muchos de estos desarrollos son aplicaciones relacionadas con muy variados dominios (ciencias aplicadas, banca, telecomunicaciones, etc.) orientadas a internet y que podrán tener consecuencias directas sobre la necesaria adaptación de los servicios de información. Por ejemplo: la aplicación para el control de la denominada firma digital (*xml digital signature*), la que relaciona *edi* (*electronic data interchange*) y xml, *vector markup language* (*vml*), la especificación *educom instructional management systems project metadata* (*ims*), *chemical markup language* (*cml*), *bioinformatic sequence markup language* (*bsml*), *weather observation markup format* (*omf*), *open financial exchange* (*ofx/ofe*), las directrices que relacionan xml y *vrml* (*virtual reality modeling language*), etc. (ver *Xml-applications* en la Bibliografía).

Otros proyectos están específicamente vinculados con los servicios de información. Naturalmente, muchas de las aplicaciones sgml han sido adaptadas a xml: se ha creado una *DTD TEI Lite* en xml (ver *TEI-Lite*), y se está desarrollando una versión xml completa de la *DTD TEI*; dentro del ámbito *TEI*, además, se ha avanzado mucho en la cuestión de los hiperenlaces (ver *TEI-extended pointers*); el modelo *EAD*, que comenzó a diseñarse en sgml, en sus últimas versiones incluye mecanismos que posibilitan la adaptación a la especificación xml.

Por otra parte, también hay proyectos que previsiblemente harán mejorar el tratamiento y distribución de la información en la Red como (entre muchos otros): el denominado *virtual hyperglossary* (*vhg*); una especificación para el intercambio de información, el *information and content exchange* (*ice*); los c*oncep-*

tual knowledge markup languages (*ckml*); un modelo xml para periódicos, el *news markup language* (*nml*) del *American Press Institute*; las comunicaciones y trabajos que se envían a las conferencias de la *Graphic Communications Association* (*GCA*) deben seguir directrices xml, de acuerdo con la *DTD "paper"*; el *theological markup language* (*thml*) que ha sido diseñado, sobre la base xml, para su uso en la denominada *Christian Classics Ethereal Library* (*Ccel*); la *data documentation initiative* (*ddi*), un esquema metadatos específico para documentación en ciencias sociales; o el proyecto para el archivo de datos lingüísticos, sonoros y textuales *Lacito* (*Linguistic data archiving project*) del *Cnrs* francés.

Esta proliferación de aplicaciones xml no significa que éste vaya a sustituir a sgml. En principio, mantener una o varias bases de datos sgml y filtrarlas a xml para las operaciones de indización, otros tratamientos de los datos y distribución en la web, se conforma como una estrategia a largo plazo con una gran probabilidad de éxito. De esta forma se cuenta con las enormes posibilidades de sgml siendo posible, al mismo tiempo, manipular y distribuir la información de forma más simple y empleando herramientas software menos complejas. Por ejemplo, la mayoría de las denominadas bibliotecas virtuales mantienen sus datos en sgml, pero distribuyen la información en xml.

Aproximación a la sintaxis sgml

La norma no determina conjuntos de códigos que se deban emplear para etiquetar estructuralmente un texto. Lo que hace es proporcionar las herramientas que permiten diseñar las instrucciones para que una máquina *entienda* un documento por su definición estructural.

Los lenguajes informáticos suelen agrupar en un segmento preliminar llamado *heading* (cabecera) un conjunto de instrucciones previas que utilizará el programa para *entender* el empleo concreto que se ha hecho de ese lenguaje. De forma similar, para procesar un documento sgml es necesario ofrecer a la máquina lo que se podría denominar como prólogo sgml y el modelo de documento concreto. El primero constará de una declaración sgml (que especifica datos básicos sobre el dialecto sgml que se ha usado, tal como el conjunto de caracteres, los códigos usados por los delimitadores) y una *DTD* (que describe con precisión los elementos que son necesarios en su elaboración y, por consiguiente, de aquellos estructurados de manera similar).

```
<!sgml "ISO 8879:1986"
      charset
    baseset "ISO 646-1983//charset International
Reference Version
(IRV)//ESC 2/5 4/0"...>
<!doctype mensaje system  "c:\xml\mensaje.dtd">
```

En este ejemplo de prólogo muy simplificado, la declaración sgml indica que se emplea su norma y se expresan, resumidos en el ejemplo, datos concretos del conjunto de caracteres empleado (*charset*) y otros códigos usados. El segundo segmento informativo indica que la *DTD* (tipo de documento, que en este caso se denomina *"mensaje"*) se halla en el "sistema" en un archivo denominado *"mensaje.dtd"*.

El documento concreto, que se denomina "modelo de documento" será un texto etiquetado. Las etiquetas utilizan delimitadores para ser distinguidas del texto en sí. Los delimitadores son *"<"* (principio de etiqueta) y *">"* (final). Normalmente una de inicio indicará el comienzo de un elemento, y una etiqueta final (añadiendo una barra inclinada al inicio de ella *"</"*) determinará su terminación (decimos "normalmente" ya que el final de un elemento puede venir indicado por el inicio del siguiente, en cuyo caso se pueden aplicar reglas de minimización).

«Actualmente la mayoría de los analizadores (parsers) se están escribiendo como applets de Java para facilitar su integración en los productos existentes»

Como se ha apuntado, las etiquetas indican la aparición de un elemento. Elemento (*element*) es la unidad estructural simple sgml. Cada uno deberá ser definido o declarado atendiendo a su contenido estructural. La declaración de elemento especificará su nombre (identificador genérico) y su modelo de contenido (es decir, qué elementos lo pueden constituir). Además de presentar jerárquicamente las declaraciones de todos los elementos que puede contener el tipo de documento en cuestión, la *DTD* incluirá las declaraciones de atributos que pueden tomar esos elementos, las declaraciones de entidad y las reglas que rigen las interacciones entre los elementos. En consecuencia, una representación gráfica imaginaria de la *DTD* sería similar a un árbol genealógico, iniciándose en un ancestro (componente superior) que será la declaración del tipo de documento y descendiendo hasta los elementos más inferiores, cuyo modelo de contenido ya no serán otros elementos, sino cualquier cadena de caracteres.

Para todo ello, sgml utiliza una serie de caracteres basada en el estándar ascii, reconocido de manera prácticamente universal por cualquier tipo de plataforma y de sistema informático. Aquellos de carácter especial, que no están contemplados en dicho conjunto (propios de sistemas de escritura distintos del inglés,

símbolos matemáticos, etc.), se transforman en representaciones ascii (utilizando el mecanismo de cadenas de sustitución) y se denominan referencias de entidad.

Utilizando algunos caracteres de forma especial, la sintaxis sgml permite especificar en los modelos de contenido si un elemento puede aparecer una o más veces (indicadores de aparición), en qué orden (conectores de grupo) y si ese modelo de contenido incluye no sólo elementos sino también grupos de ellos (modelo de grupo). Aquellas características especiales de los elementos (como por ejemplo que uno concreto pueda tomar diferente valor dependiendo de las circunstancias) que no deben ser incluidas en las declaraciones de elemento se reflejan declarando los denominados atributos de elemento.

Hay otras posibilidades sintácticas (la inclusión o exclusión de elementos en determinadas condiciones o las estructuras concurrentes) que también contempla la norma. Más importantes son las denominadas *entities,* citadas anteriormente, que no sólo sirven para representar un determinado carácter, sino que también pueden emplearse para hacer referencias, expansiones y conexiones a ficheros concretos, sistemas o enlaces internos o externos.

El "prólogo" anteriormente propuesto *enviaría* al sistema al siguiente archivo que presenta la aclaración de tipo de documento *"mensaje*", que podría corresponder a un mensaje electrónico. En este caso, la *DTD* (ya se ha comentado que podría estar integrada en un archivo independiente) incluiría las aclaraciones de los elementos estructurales que conformarían el tipo de documento "mensaje electrónico".

```
<!doctype mensaje [
<!element mensaje - - (sobre,texto)>
<!element sobre - o (para,cc,cco,asunto)>
<!element para - o (dir+)>
<!element cc - o (dir*|empty)>
<!element cco - o (dir*|empty)>
<!element asunto - o (#pcdata|empty)>
<!element dir - - (user,arroba,dom)>
<!element user - o (#pcdata)>
<!element dom - o (#pcdata)>
<!element texto - - (#pcdata&attach*)>
<!element attach - - (#pcdata)>
<!element attachref - - (empty)>
<!entity attach system "attachref">
<!entity arroba "&#122;">
<!attlist attach id id #required>
<!attlist attachref target idref #required>
          ]>
```

Las aclaraciones de elementos incluyen: el identificador genérico (su nombre), los códigos de minimización correspondientes a la etiqueta inicio y a la de final ("-" para obligatoria y "O" para opcional) y los modelos de contenido (qué elementos pueden componer a aquellos que les incluyen). En estos modelos de contenido se incluyen otros elementos (que luego deberán ser aclarados) o los elementos "finales" —que son prefijados como "*#pcdata*" (cualquier cadena de caracteres) o *"empty*" (vacío)—, con sus indicadores de aparición —"+" (una o más veces), *"?*" (una o ninguna), *"*" (ninguna, una o más)— y los denominados conectores —"," (en ese orden), "|" (sólo uno de ambos), "&" (ambos pero en cualquier orden)—. Esta *DTD* ejemplo también incluye las aclaraciones de entidades (de sistema para "enviar", de *"attach*" a *"attachref*" y de carácter para representar *"arroba*") y las aclaraciones de atributos (la identidad de *"attach*" que será un *"Id*" en cada caso y es obligatorio, y el "puntero referencia" a dicho *"attach*").

Software específico

Cada vez es mayor la cantidad disponible para el procesamiento sgml. Ya sea comercial o público, el corazón de este software es el denominado *parser* o analizador. Para extraer todas las posibilidades de flexibilidad e independencia de sgml, es imprescindible que el documento se ajuste perfectamente a la *DTD* que le corresponde. En caso contrario el sistema no comprendería dicho documento. El analizador, por lo tanto, es una parte de software que toma una *document instance*, lee la *sgml declaration* (que le dice al sistema que se trata de un documento sgml) y utiliza la *doctype declaration*, para saber de qué tipo de documento se trata y cuál es su *DTD*. Este proceso culmina con la equiparación entre la *DTD* y el modelo de documento concreto que se está procesando. Aunque, en términos simplistas, el producto de un analizador es "sí" (el modelo de documento es válido) o "no" (no lo es), hay muchos tipos de *parser* con funcionalidades muy distintas. Actualmente la mayoría de los analizadores se están escribiendo como *applets* de *Java* para facilitar su integración en los productos existentes.

Sobre este corazón se desarrollan herramientas sgml cada vez más poderosas. Se dispone ya de aplicaciones para la edición, creación y procesamiento en diferentes versiones, y los principales distribuidores de software comienzan a proporcionar extras sgml a sus productos. En estos momentos, en concreto, el mayor crecimiento en el mercado de software genuino sgml corresponde a herramientas que permiten la fácil transformación y conversión de documentos, aplicaciones para el visionado sgml y software de gestión documental (**Pepper**; *Sgml/xml-software*).

Un editor podría asemejarse a una especie de procesador de textos inteligente. Puede usar información extraída de una *DTD* procesada para informar al usuario sobre qué elementos se requieren en diferentes

puntos de un documento cuando está siendo creado (de hecho, muchos de los editores funcionan como compiladores). También puede simplificar enormemente la tarea de prepararlo, por ejemplo, insertando etiquetas automáticamente.

Las aplicaciones que se podrían denominar conversores permiten aprovechar la flexibilidad sgml tantas veces aludida en este trabajo. La conversión, normalmente, implica la construcción de un puente entre el mundo de los documentos impresos y de procesadores de textos (en los que la estructura lógica es percibida visualmente por el lector) y los documentos "inteligentes" (en los que la estructura lógica está explícitamente codificada) (**Severson**, 1995). Este proceso puede tener muchos destinos: html, *postscript* para imprimir directamente, texto plano, *LyX* (*frontend wysiwyg* para *LaTeX*), *rtf* (*rich text format*), etc.

«A pesar de su nombre, html representa una minúscula parte de la funcionalidad que históricamente ha estado asociada con el concepto de sistemas hipertexto»

Un formateador opera sobre un modelo de documento etiquetado para producir un formato impreso de él. Muchas distinciones tipográficas, como el uso de fuentes o tamaños concretos, están íntimamente relacionadas con diferenciaciones estructurales. Los formateadores pueden, por lo tanto, obtener ventaja a través del marcaje descriptivo. Sin embargo, hoy día la gran mayoría de formateadores emplean una definición de formato en términos sgml. El programa ayuda al diseño de una *DTD* específica para la presentación, basada en *dsssl*, aplicándola posteriormente. Algunos sistemas, como *Adept* de *Arbortext*, posibilitan el filtrado a xml y, consiguientemente, la elaboración de hojas de estilo *xsl*.

Respecto a la gestión documental, están siendo desarrollados actualmente muchos productos para extender las posibilidades de sistemas de bases de datos (textuales y no textuales), aprovechando la información estructurada explicitada mediante marcaje sgml. Sin embargo es clara la necesidad de nuevas familias que integren plenamente los productos xml.

El acomodo: xml, *xsl*, *xlink*

Xml fue creado, precisamente, debido a que las alternativas, html y sgml no son útiles para este propósito. Así pues, las principales adaptaciones que presenta xml con respecto a sgml son (**Clark**, 1997):

—Para procesar muchos documentos xml no es necesaria una *DTD*,

—los "modelos de documento" pueden incluir instrucciones de procesado,

—en los modelos de contenido de las declaraciones de elementos no se puede emplear el conector *"&*", no se permite usar las *"exception*" y no se utilizan los indicadores de minimización ya que, normalmente, siempre se requiere la aparición de ambas etiquetas,

—se admite el uso generalizado de elementos "vacíos",

—se emplean las denominadas secciones *"cdata*" y las *notation declarations* (declaraciones de anotación), entre otros nuevos componentes,

—a menos que se indique lo contrario, los espacios en blanco, cambios de línea o "retornos" son significativos, y

—se usa el lenguaje *ebnf* (*extended backus-naur form*) para expresar la sintaxis xml.

De las características citadas, entre muchas otras, quizá la más importante sea la primera. Efectivamente, muchos documentos xml pueden ser procesados sin una declaración, circunstancia que puede simplificar enormemente la complejidad de las aplicaciones para el procesamiento. Esto implica que pueden existir dos categorías de documentos xml: los denominados *well-formed* (bien formados) y los válidos. Los primeros son aquellos que pueden ser procesados a pesar de no presentar una *DTD*, ya que obedecen rígidamente la sintaxis xml (la denominada especificación xml). Algunos documentos (los válidos) además de obedecer a dicha especificación, deben responder a una *DTD* (sobre todo en contextos de producción) o a una parte de ella (con relación a los valores por defecto de los atributos o con el uso no significativo de los espacios en blanco sobre todo). Si es esto lo que ocurre, la referencia a la declaración del tipo de documento debe ser el primer componente del documento, tras las opcionales "instrucciones de procesado" (además de eventuales "comentarios", que no se procesarían como formando parte del documento).

La declaración del tipo de documento identifica el elemento raíz del documento y puede contener declaraciones adicionales. Todos los documentos xml deben tener un elemento raíz simple que abarque la totalidad de su contenido. Estas declaraciones complementarias pueden proceder de una definición "externa" (una *DTD*), ser incluidas directamente en el documento o las dos cosas. Quiere esto decir que para muchos procesadores xml no es necesaria la validación del documento (su confrontación con una *DTD* completa) si no

se explicita esta necesidad. Las posibles modificaciones a la especificación general (relativas sobre todo a los atributos, los enlaces o el uso de los espacios en blanco) pueden ser incluidas en una declaración "interna" al propio documento.

Para expresar la necesidad de validación del documento se emplea el segmento semántico denominado *required markup declaration* (declaración de etiquetado exigida) o *RMD*. Un valor *rmd="internal*" advierte que sólo será necesario procesar las declaraciones internas, *"all*" indica que deben serlo tanto la interna como la externa, y un valor *"none*" que el documento puede ser procesado sin leer ninguna de ellas.

Normalmente un documento xml (W3C-xml) comienza con una instrucción de procesado (*Pi*) que toma la forma: *<?xml version="1.0"?>*. Su presencia identifica el documento que sigue como xml e indica la versión concreta utilizada. Como los denominados comentarios, las *Pi* no forman parte del texto del documento. Si es necesaria su presencia, a esta *Pi* seguirá la *rmd*, tomando, por ejemplo, la forma: *<?xml version="1.0" rmd="internal"?>*.

Los modelos de contenido de las declaraciones de elementos no emplean el conector *"&*", ya que los elementos deben aparecer siempre en el orden especificado. No se permite el uso de las excepciones a los modelos de contenido, ya que un elemento siempre tendrá como tal el especificado y ningún otro. En un modelo de documento los elementos siempre estarán perfectamente delimitados por su etiqueta de inicio y de final, por lo que no es necesaria la aparición de los códigos que representan las reglas de minimización.

Los elementos vacíos tienen una sintaxis modificada, como por ejemplo *<nombre/>*, para indicar al procesador que no tiene contenido y por lo tanto no es necesaria la aparición de la etiqueta final. Los vacíos suelen emplearse para indicar que algo ocurre (por ejemplo una referencia cruzada). También se puede utilizar, como en sgml, el modelo de contenido *"empty*" o el nuevo *"any*", que indica que puede ser permitido cualquier contenido. El uso de este último modelo de contenido, por la peligrosa ambigüedad que representa, está casi exclusivamente restringido a los procesos de conversión.

Las secciones *"cdata*" se usan para prevenir que caracteres como *"<*" o *"&*", que podrían formar parte del texto, sean considerados por el *parser* como "marcas". Toman, por ejemplo, la forma: *3 <![cdata[<]]> 4*. Todos los datos de carácter que se encuentren entre el principio de la sección *"<[cdata[*" y su final *"]]>*" son pasados directamente a la aplicación. Los denominados comentarios y el conjunto de caracteres *"]]>*" son las únicas cadenas que no pueden aparecer en una sección *cdata*.

Las *notation declarations* son empleadas para identificar tipos concretos de datos binarios externos (por ejemplo identificación de un archivo gif: *<!notation gif3 system "gif">*).

En xml los espacios son separadores (*lf*, *ret*, *tab*). Como se ha dicho, en algunos casos el espacio en blanco es significativo y, además de los separadores, se puede emplear el atributo *"xml:space*".

«Sgml es una norma clave que jugará durante mucho tiempo un papel vital en la creación de nuevos servicios de información a usuarios finales»

Ebnf es un conjunto de reglas denominadas *productions* y su uso para expresar la sintaxis de xml facilita su manejo con las modernas herramientas de compilación. Cada regla describe un fragmento específico de sintaxis. Un documento será válido si puede ser reducido a una única y específica regla. A pesar de que *ebnf* no es una eficaz forma de representar sintaxis para uso humano, existen programas que pueden volcar automáticamente *ebnf* en un *parser*.

Como su norma matriz, xml está especialmente indicado para representar estructuras. Precisamente una de sus principales características es que separa completamente la estructura del contenido de un documento de su presentación concreta. Esto constituye una de sus principales ventajas, ya que es posible modificar la visualización repetidamente sin tener que tocar el contenido. Para aplicar formatos de forma estandarizada a los documentos xml se emplea otra especificación denominada *xsl*.

Xsl es un lenguaje para definir hojas de estilo aplicables a los documentos xml y se basa en el estándar *ISO dsssl*, que es también la base de otros sistemas de definición de estilos como *css2*, el aplicado en html. De hecho, *xsl* incluye la mayoría de objetos de formato y propiedades de *css*.

Dado un tipo de documentos estructurados o ficheros de datos en xml, un diseñador puede usar una *stylesheet* (hoja de estilo) *xsl* (ver *W3C-stylesheet*) para expresar sus intenciones sobre cómo se debe presentar este contenido estructurado; es decir, qué estilo se debe aplicar al contenido fuente, cómo se debe distribuir y paginar sobre algún medio de presentación (una ventana en un buscador web o un conjunto de páginas físicas en un libro).

Este proceso de presentación, definido por la especificación *xsl* (ver *W3C-xsl*), está formado por dos subprocesos: el primero de ellos, construir lo que se podría denominar como árbol resultado *(result tree)* desde el árbol fuente xml y, el segundo, interpretar el árbol resultado para producir una presentación formateada. El primer sub-proceso se denomina *tree transformation*, y el segundo es llamado *formatting*.

Esta división proporciona una gran flexibilidad al construir la presentación del contenido fuente. Con la *tree transformation* la estructura del árbol resultado puede ser totalmente diferente de la estructura del árbol fuente. Una hoja de estilo contendrá un conjunto de *tree construction rules*. Estas reglas tendrán dos partes: una estructura que se corresponderá con los elementos del árbol fuente y un esquema que constituirá una sección del árbol resultado. Esto permitirá que una hoja de estilo sea aplicable a una amplia variedad de documentos que tengan estructuras similares de árbol fuente. El mecanismo de la *tree construction* está descrito en una parte de la recomendación *xsl*, considerada como normativa y denominada *xslt* o *xsl transformation* (ver *W3C-xslt*).

El segundo proceso, el formateo, permite dotar de una semántica de formato al árbol resultado y expresada mediante un catálogo de *formatting objects* (objetos de formato). Los nudos del árbol resultado son modelos de objetos de formato, los cuales representan abstracciones tipográficas como página, párrafo, etc. Un conjunto de *formatting properties* —propiedades de formato como sangrado, espaciado, etc.— controlan la presentación de estas abstracciones. La conjunción entre los objetos de formato y sus propiedades de formato proporciona el vocabulario que expresa la presentación en términos xml.

La concreción del formato comienza cuando el árbol resultado ha sido completamente construido usando las reglas de construcción de árbol. Sintácticamente, un modelo de objeto de formato está representado como un elemento xml, con las propiedades determinadas mediante un conjunto de pares atributo-valor. El contenido del modelo de objeto de formato es el propio del elemento. El vocabulario de los objetos de formato aportados por *xsl* (el conjunto de elementos del tipo *"xsl:fo"*) constituye el conjunto de unidades de organización tipográficas disponibles para el diseñador. Semánticamente, cada modelo de objeto de formato representa una especificación para una parte de la información de paginación, presentación y estilo que será aplicada al contenido de ese objeto de formato. Cada clase de objeto de formato representa un tipo particular de comportamiento de formato.

Las propiedades asociadas con un modelo de objeto de formato controlan el formato de este objeto. Algunas de las propiedades, por ejemplo "color", especifican directamente el resultado, otras, como el control de guiones, tan sólo limitan el conjunto de posibles resultados de configuración sin especificar cualquier resultado concreto.

El siguiente ejemplo es una pequeña parte de una *DTD* que representa la estructura de un árbol resultado que contiene objetos de formato y propiedades aplicables a los mismos.

```
<!—
      Declaración de entidad para grupos de
objetos de formato
—>
<!entity % block-level "
   fo:block
 | fo:display-sequence
 | fo:display-graphic
 | fo:display-included-container
 | fo:display-rule
 | fo:table-and-caption
 | fo:list-block
 | fo:table
 | fo:multi-switch
 | fo:multi-properties
">
```

En este caso se define un grupo de objetos de formato denominado *"block-level"* que se usa normalmente para proporcionar formato a párrafos, títulos, encabezados, etc. En realidad es un grupo de objetos de formato relacionados porque representan áreas (bloques) de contenido.

A cada uno de los objetos de formato de *"block-level"* le corresponde una declaración de elemento y una declaración de la lista de atributos (que representan propiedades de formato) que pueden afectar a dicho elemento objeto de formato. En concreto, para el *"fo block"*, propiamente dicho, serían:

```
<!—
      Estructura de elementos y atributos
—>

<!element fo:block (
 fo:first-line-marker?,
   (
    #pcdata
   | %inlines;
   | %block-level;
   )+
 )
>

<!attlist fo:block
```

```
    %common-absolute-position-properties;
    ...
    %common-font-properties;
    ...
    writing-mode  cdata  #implied
>
```

Previamente, han sido declarados los grupos de propiedades. Para el caso de *"common-absolute-position-properties*", por ejemplo, la declaración sería la siguiente:

```
<!—
      Declaración de entidad para grupos de
propiedades
—>

<!entity % common-absolute-position-properties "
     bottom  cdata  #implied
     left  cdata  #implied
     position  cdata  #implied
     right  cdata  #implied
     top  cdata  #implied
">
```

Está definido el documento xml y se le puede proporcionar una presentación. Falta algo sin lo que su presencia en la Red no tendría sentido: los hiperenlaces.

A pesar de su nombre, html representa una minúscula parte de la funcionalidad que históricamente ha estado asociada con el concepto de sistemas hipertexto. Por poner un ejemplo: tan sólo hace posible la forma más simple de enlaces (unidireccionales y especificando exclusivamente la localización del recurso al que apuntan).

Un verdadero sistema hipertexto debe establecer una sintaxis estandarizada para todos los mecanismos clásicos de enlaces: denominación independiente de la localización; vínculos bidireccionales y que puedan gestionarse y especificarse desde fuera de los documentos a los que se apliquen; hiperenlaces múltiples (*N-ary*): anillos, ventanas múltiples; enlaces agrupados (múltiples orígenes); que permitan la transclusión: el documento destino (al que apunta la relación) aparece como parte del documento origen del enlace y atributos en los enlaces (tipos de enlaces). Todo esto es lo que se pretende conseguir con los mecanismos de unión y localización xml.

Los dispositivos *xml linking and addressing* se describen en tres especificaciones del *W3C*: *xpath* (W3C-xpath), *xpointer* (W3C-xpointer) y *xlink* (W3C-xlink) propiamente dicho.

Xlink es una especificación que define construcciones que pueden ser insertadas en *DTD*s y modelos de documento xml para describir enlaces entre recursos. En concreto, define la sintaxis (en términos xml) que debe emplearse para mostrar la existencia de un(os) enlace(s) y describir sus características.

Un caso simple de establecer una relación de enlace en un documento xml es el mecanismo *Id/Idref*. Sin embargo, *xlink* proporciona un sistema que extiende esta capacidad básica: puede representar relaciones entre múltiples recursos, emplea un dispositivo explícito para asociar metadatos con el enlace y aporta el valor añadido de los enlaces *out-of-line*. *Xlink* define los enlaces teniendo en cuenta una serie de importantes características: sintaxis del localizador, reconocimiento de los vínculos, sus atributos y sus elementos de enlaces.

La sintaxis del localizador se suministra mediante una *URI-reference*, que es un *URI* (*uniform resource identifier*, *Ietf RFC 2396*) con un fragmento identificador opcional separado del mismo por un carácter *"#"*. Para localizadores dentro de recursos xml el formato del fragmento identificador viene definido por la especificación x*pointer*.

Podrían contemplarse varias formas para reconocer los enlaces. Sin embargo *xlink* propone el uso de los nombres de elementos y atributos dentro del *xlink namespace* asociado con el siguiente *URI*: *http://www.w3.org/TR/REC-xml-names*. Formando también parte de la que se podría llamar familia xml, *xml Namespaces* (W3C-namespaces) es una especificación que ofrece instrucciones que permiten distinguir cada elemento en un entorno específico (pueden darse distintos usos de un mismo elemento dependiendo del contexto). En el caso de *xlink*, los atributos de enlace requerirán el prefijo *xlink namespace* si aparecen dentro de un elemento cualquiera o pueden ser usados directamente si existen en un elemento *xlink* explícito.

Xlink tiene varios atributos asociados con la variedad de enlaces que se pueden representar y que definen cuatro conceptos principales: *locators*, *arc ends*, *behaviors* y *semantics*, para concretar valores relativos respectivamente a: la localización del recurso, el *recorrido* del enlace, el modo en que es activado y lo que debe hacer la aplicación con el recurso que está siendo enlazado, e información útil (título, por ejemplo) que puede usar la aplicación.

Hay varios tipos de elementos de enlace en *xlink*: *simple links*, *locators*, *arcs*, *extended links*, *extended links group* y *extended link group documents*. El enlace simple se usa para declarar un vínculo funcionalmente parecido al elemento *"A"* de html, pero que presenta características añadidas (como semántica y comportamiento) que incrementan su valor. Los elementos *locator* se usan para definir el recurso que está siendo enlazado. Algunos enlaces pueden contener localiza-

dores múltiples, representando la variedad de enlaces que pueden ser activados. Los *arcs* se usan para determinar la semántica del recorrido del enlace. Un elemento de enlace *extended* difiere de un vínculo simple en que puede conectar cualquier número de recursos y que, a menudo, son del tipo *out-of-line* (no se hallan en el recurso a que hacen referencia).

Xlink, como se apuntó anteriormente, incrementa el valor de los enlaces simples, como en el siguiente ejemplo:

```
<estudiante id="Antonio"
xmlns:xlink="http://www.w3.org/xml/xlink/0.9">
      <nombre>Antonio</nombre>
<profesor xlink:href="porfesores.xml#Edu"
xlink:title="tutor" xlink:show="parsed"
xlink:actuate="auto"/>
</estudiante>
```

Además de expresar la localización, se añade información descriptiva como el título del recurso e información *de comportamiento* (se mostrará como formando parte del documento desde el que se activa *"parsed*" y se hará de forma automática *"auto*").

Pueden presentarse enlaces multidireccionales que posibilitan, por ejemplo, la vuelta al recurso inicial utilizando el vínculo de "vuelta" (con un enlace bidireccional):

```
<xlink:extended
xmlns:xlink="http://www.w3.org/xml/xlink/0.9">
      <xlink:locator href="#Antonio"
id="estudiante"/>
      <xlink:locator href="profesores.xml#Edu"
id="profesor"/>
      <xlink:arc from="estudiante" to="profesor"
show="parsed"/>
</xlink:extended>
<estudiante id="Antonio">
      <nombre>Antonio</nombre>
</estudiante>
```

Se puede advertir que se trata de un enlace extendido que presenta el elemento *"from*" dentro del propio documento.

Pueden darse, también, otros con múltiples destinos (el usuario puede elegir entre varios desde un único enlace). Como se ha visto, es posible que el enlace se active sin intervención del usuario y que el contenido del recurso enlazado se integre en el documento fuente. Además el contenido del documento que se está consultando puede ser reemplazado por el del propio recurso activado, y es posible organizar y gestionar los enlaces desde alguna base de datos.

Xlink desarrolla el mecanismo para diseñar e insertar enlaces en documentos xml, y que pueden apuntar a cualquier tipo de fuente. Era necesario, no obstante, especificar un lenguaje que permitiese enlazar a un documento xml (o parte del mismo) desde cualquier parte. Esto último se consigue con *xpointer*, que utiliza la gramática *xpath*.

Xpath es una especificación que proporciona una sintaxis y una semántica comunes a *xsl transformation* (*xslt*) y *xpointer*, un lenguaje para descubrir partes de un documento xml, además de para poder manipular cadenas, números y expresiones booleanas. Usa una sintaxis compacta no-xml que facilita su uso en *URI*s y valores de atributo xml. Opera en abstracto, sobre la estructura lógica de un documento xml, más que en su superficie sintáctica. Recibe el nombre de *xpath* porque usa una notación *path* (como la de los URL) para navegar sobre la ordenación jerárquica de un documento xml. Además de ser usado para localizar, también puede utilizarse para la equiparación (comprobando si un nodo es o no equiparable a su modelo); este último uso de *xpath* es descrito en *xslt*.

Xpath modela un documento xml como un árbol de nodos. Hay diferentes tipos de nodos, incluyendo los de elemento, de atributo y de texto. *Xpath* define una forma de calcular un *string-value* (valor-cadena) para cada una de estas clases. Algunos tipos de nodos también tienen nombres basados en *xml namespaces*. El nombre de un nodo tendrá una parte interna y posiblemente un *namespace URI* nulo; esto es lo que se denomina un nombre expandido.

Xpointer define un lenguaje para la localización dentro de las estructuras internas de documentos xml. En concreto, ofrece el medio de especificar referencias a elementos, cadenas de caracteres y otras partes de documentos xml, tengan o no un atributo *Id* explícito. *Xpointer* utiliza el lenguaje de expresiones *xpath* y lo aplica extendiendo su uso al permitir localizar rangos además de nodos, localizar información por medio de la equiparación de cadenas y usar expresiones de localización en *URI*s como fragmentos identificadores. Las expresiones *xpointer* encuentran información navegando a través de la estructura de un documento y seleccionando partes basadas en propiedades como tipos de elementos, valores de atributo, contenido carácter y orden y posición relativos. Proporciona medios para identificar localizaciones en documentos xml, independientemente de si han de servir como destinos de enlace o para cualquier otro propósito.

Los trabajos sobre la especificación *xua* (*xml user agent*) aún no han comenzado, pero cuando se desarrolle permitirá estandarizar el uso que las aplicacio-

nes cliente (navegadores, etc.) harán de los documentos xml.

Conclusiones

Con la tendencia creciente hacia la publicación electrónica, los modelos tradicionales de producción, distribución y uso de información científica están siendo reevaluados por los editores y los servicios de información. Desde esta perspectiva, sgml es una norma clave que jugará durante mucho tiempo un papel vital en la creación de nuevos servicios de información a usuarios finales.

Su uso se presenta como una elección obvia para estos servicios debido a que posibilita un tratamiento informático de los datos adecuado y no excesivamente complejo; puede ser fácilmente aplicable a las descripciones bibliográficas, debido a que esta información contiene texto y posee una estructura lógica rigurosa (las reglas de catalogación), adecuada para ser descrita en una *DTD* y ofrece flexibilidad para incluir elementos descriptivos *Marc* y elementos no-*Marc*. Es más, usando la técnica de referencias de entidad, sgml atiende a solución de problemas relacionados con caracteres especiales y tiene características predeterminadas que, mediante el uso de aplicaciones específicas (*parsers*), permiten el control de calidad de los datos entrantes.

Su capacidad para la estructuración lógica y la versatilidad de diseño la convierten en una norma ideal para desarrollar esquemas de metadatos. Su capacidad discriminatoria maximiza las posibilidades de indización y recuperación. Por otra parte, la separación de la estructura del documento de su estilo de representación concreto abre la posibilidad de volver a usar los documentos sgml para diferentes propósitos (entorno de red, bases de datos en línea o cd-rom) y con distintos formatos de presentación para el usuario final (html, formatos privados, lenguajes de etiquetado orientados específicamente a copias impresas como *rtf*, *LaTeX*, *Postscript*, etc.).

No obstante, la política con respecto a sgml en los servicios de información se ha desarrollado sólo muy recientemente debido, sobre todo, al alto coste (en recursos materiales, humanos, de formación, etc.). Por lo tanto, para estos sistemas podría ser importante iniciar pequeños proyectos sgml, con alta visibilidad para los usuarios finales, lo que mejoraría su satisfacción y, previsiblemente, permitiría ampliar a largo plazo las inversiones.

Para explotar las posibilidades de la web, sgml no es adecuado. Aunque html es esencialmente una *DTD* sgml, al ser limitado y preestablecido, no permite disponer de muchas de sus funcionalidades. La implementación de aplicaciones sgml en la Red es prácticamente inviable. En este contexto, xml y las especificaciones complementarias se presentan como una herramienta eficaz para la distribución de documentos electrónicos en internet.

Esto no quiere decir que xml pueda sustituir a sgml. A pesar de que xml está diseñado para presentar documentos estructurados en la web, alguna de las características de las que carece hacen que sgml sea una solución mucho más práctica para la creación y almacenamiento a largo plazo de documentos complejos.

Los servicios de información podrán obtener claras ventajas usando estos estándares tecnológicos. En términos simplistas, la aplicación podría consistir en mantener una o varias bases de datos con todo tipo de objetos informativos en formato sgml y manipular esa información de forma casi instantánea para producir otros objetos informativos elaborados ad hoc y distribuir esta información a través de la Red o presentarla sobre cualquier medio mediante el filtrado a xml.

Bibliografía

Adams, R. J. "Electronic libraries sgml applications: background to project Elsa". En: *Program*, 1995, v. 29, n. 4, pp. 397-406.

Adler, S. C. "The «ABCs» of dsssl". En: *Journal of the American Society for Information Science*, 1997, v. 48, n. 7, pp. 597-602.

Adler, S. C. "The birth of a standard". En: *Journal of the American Society for Information Science*, 1992, v. 43, n. 8, pp. 556-558.

Baird, S.; **Wu, G.**; **Robinson, B.** "HyTime: hypermedia/time-based structuring language". *British Library Research and Development Department. Blrd-report* 6156, 1994.

Burnard, L.; **Light, R.** Three sgml metadata formats: TEI, EAD, and Cimi: a study for Biblink work package 1.1, 1996. Consultado en: 25-02-00. *http://hosted.ukoln.ac.uk/biblink/wp1/sgml*

Clark, J. Comparison of sgml and xml, 1997. Consultado en: 25-02-00. *http://www.w3.org/TR/NOTE-sgml-xml.html*

Cole, T. W.; Kazmer, M. M. "Sgml as a component of the digital library". En: *Library hi tech*, 1995, v. 13:4, n. 52, pp. 75-90.

Corthouts, J.; Philips, R. "Sgml: a librarian's perception". En: *The electronic library*, 1996, v. 14, n. 2, pp. 101-110.

DeRose, S. J. "Navigation, access, and control using structured information". En: *American archivist*, 1997, v. 60, n. 3, pp. 298-309.

Dunlop, D. "Practical considerations in the use of TEI headers in large corpora". En: *Computers and the humanities*, 1995, v. 29, n. 1, pp. 85-98.

Entlich, R.; Garson, L.; Lesk, M.; Normore, L.; Olsen, J.; Weibel, S. L. "Making a digital library: the chemistry online retrieval experiment". En: *Communications of the ACM*, 1995, v. 38, n. 4, p. 54.

Fausey, J.; Shafer, K. "All my data is in sgml. Now what?". En: *Journal of the American Society for Information Science*, 1997, v. 48, n. 7, pp. 638-643.

Gartner, R. "Digitising the Bodleian revisited: linking word and image". En: *Audiovisual librarian: multimedia information*, 1997, v. 23, n. 1, pp. 47-50.

Giordano, R. "Documentation of electronic texts using text encoding initiative headers: an introduction". En: *Library resources and technical services*, 1994, v. 38, n. 4, pp. 5-27.

Goldfarb, C. F. *The sgml handbook*. Oxford: Oxford University Press, 1990.

Goldfarb, C. F. "Sgml: the reason why and the first published hint". En: *Journal of the American Society for Information Science*, 1997, v. 48, n. 7, pp. 656-661.

Heath, L. S.; Hix, D.; Nowell, L. T.; Wake, W. C.; Averboch, G. A.; Labow, E.; Guyer, S. A.; Brueni, D. J.; France, R. K.; Dalal, K.; Fox, E. A. "Envision: a user-centered database of computer science literature". En: *Communications of the ACM*, 1995, v. 38, n. 4, pp. 52-53.

Hockey, S. "Evaluating electronic texts in the humanities". En: *Library trends*, 1994, v. 42, n. 4, pp. 676-693.

ISO-dsssl. *International Organization for Standardization, International Electrotechnical Commission. ISO/IEC 10179:1996. Document style semantics and specification language (dsssl)*. Geneva: ISO/IEC, 1996.

ISO-HyTime. *International Organization for Standardization. ISO/IEC 10744-1992 (E). Information technology-hypermedia/time-based structuring language (HyTime)*. Geneva: ISO, 1996. Consultado en: 25-02-00. *http://www.ornl.gov/sgml/wg8/docs/n1920/*

ISO-sgml. *International Organization for Standardization. ISO 8879-1986 (E). Information processing-text and office systems-standard generalized markup language (sgml)*. Geneva: ISO, 1986.

Larson, R. R.; McDonough, J.; O'Leary, P.; Kuntz, L. "Cheshire II: designing a next-generation online catalog". En: *Journal of the American Society for Information Science*, 1996, v. 47, n. 7, pp. 555-567.

Marcoux, Y.; Sévigny, M. "Why sgml? Why now?". En: *Journal of the American Society for Information Science*, 1997, v. 48, n. 7, pp. 584-592.

Palowitch, C.; Horowitz, L. "Meta-information structures for networked information resources". En: *Cataloging and classification quarterly*, 1996, v. 21, n. 3/4, pp. 109-130.

Pepper, S. Whirlwind guide to sgml tools and vendors. Consultado en: 25-02-00.
http://www.infotek.no/sgmltool/guide.htm

Philips, R.; Corthouts, J. "HyperLib: a hypertext interface to a library information system". En: *Library technology news*, 1995, v. 17, pp. 4-6.

Pitti, D. "Encoded archival description. An introduction and overview". En: *D-lib magazine*, 1999, v. 5, n. 11. Consultado en: 25-02-00. *http://www.dlib.org/dlib/november99/11pitti.html*

SAA. *Encoded archival description application guidelines: version 1.0*. Chicago: Society of American Archivists, 1999. Consultado en: 25-02-00. *http://www.archivists.org/catalog/index.html*

Severson, E. The art of sgml conversion: eating your vegetables and enjoying dessert, 1995. Consultado en: 25-02-00.
http://www.ileaf.com/avhome/veggies.html

Sgml/xml-software. Consultado en: 25-02-00.
http://www.oasis-open.org/cover/SW.html

Sgml/xml web page. Consultado en: 25-02-00.
http://www.oasis-public.com/cover

Sperberg-McQueen, C. M.; **Burnard, L.** *TEI P3: guidelines for electronic text encoding and interchange*. Oxford [etc.]: TEI, 1994. Consultado en: 25-02-00.
http://www-tei.uic.edu/orgs/tei/sgml/teip3sg/SG.htm

TEI-extended pointers. Consultado en: 25-02-00.
http://etext.virginia.edu/bin/tei-tocs?div=DIV2&id=SAXR

TEI-lite. TEI lite DTD in xml. Consultado en: 25-02-00.
http://www.uic.edu/orgs/tei/lite/teixlite.dtd

Van Herwijnen, E. *Practical sgml*. Dordrecht: Kluwer Academic Publisher, 1994.

W3C-css2. World Wide Web Consortium. Cascading style sheets, level 2 (css2). Recomendación 12 de mayo de 1998. Consultado en: 25-02-00. *http://www.w3.org/TR/1998/REC-CSS2-19980512*

W3C-namespaces. World Wide Web Consortium. Namespaces in xml. Recomendación 14 de enero de 1999. Consultado en: 25-02-00.
http://www.w3.org/TR/1999/REC-xml-names-19990114/

W3C-stylesheet. World Wide Web Consortium. Associating stylesheets with xml documents. Working draft. Consultado en: 25-02-00.
http://www.w3.org/TR/WD-xml-stylesheet

W3C-xlink. Xml linking language (xlink). Working draft 26 de julio de 1999. Consultado en: 25-02-00.
http://www.w3.org/1999/07/WD-xlink-19990726

W3C-xml. World Wide Web Consortium. Extensible markup language (xml) 1.0. Recomendación 10 de febrero de 1998. Consultado en: 25-02-00. *http://www.hispalinux.es/'http//www.w3.org/TR/1998/REC-xml-19980210.html*

W3C-xpath. Xml path language (xpath) versión 1.0. Recomendación propuesta 8 de octubre de 1999. Consultado en: 25-02-00.
http://www.w3.org/TR/1999/PR-xpath-19991008

W3C-xpointer. Xml pointer language (xpointer). Working draft 9 de julio de 1999. Consultado en: 25-02-00.
http://www.w3.org/1999/07/WD-xptr-19990709

W3C-xsl. World Wide Web Consortium. Extensible stylesheet language (xsl) specification. Working draft 21 de abril de 1999. Consultado en: 25-02-00.
http://www.w3.org/TR/1999/WD-xsl-19990421

W3C-xslt. World Wide Web Consortium. Xsl transformation (xslt). Working draft 21 de abril de 1999. Consultado en: 25-02-00.
http://www.w3.org/TR/199/WD-xslt-19990421.html

Weibel, S. L. "The world wide web and emerging internet resource discovery standards for scholarly literature". En: *Library trends*, 1995, v. 43, n. 4, pp. 627-644.

Xml-applications. Consultado en: 25-02-00.
http://www.oasis-open.org/cover/xml.html.

Xml FAQ. *Frequently asked questions about the extensible markup language. The xml FAQ*. **Peter Flynn**, University College Cork. Versión 1.5. Consultado en: 25-02-00.
http://www.ucc.ie/xml/

Eduardo Peis *y* ***Félix de Moya.*** *Universidad de Granada. Facultad de Biblioteconomía y Documentación.*
epeis@ugr.es
felix@ugr.es